UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS FÍSICAS

OPTIMIZACIÓN DEL DISEÑO FÍSICO DE CIRCUITOS DIGITALES ORIENTADO A DISPOSITIVOS RECONFIGURABLES

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Juan de Vicente Albendea

Bajo la dirección del doctor

Juan Lanchares

Madrid, 2004

ISBN: 978-84-669-1771-1

OPTIMIZACIÓN DEL DISEÑO FÍSICO DE CIRCUITOS DIGITALES ORIENTADO A DISPOSITIVOS RECONFIGURABLES

JUAN DE VICENTE ALBENDEA

Universidad Complutense de Madrid
Facultad de Ciencias Físicas
Tesis Doctoral

Madrid, 2001

Resumen

Los dispositivos reconfigurables están revolucionando los procesos de desarrollo y producción de sistemas digitales. Debido a la complejidad creciente de los sistemas, y para sacar el máximo partido a los recursos que ofrecen los dispositivos reconfigurables, es imprescindible la automatización de los procesos implicados en el desarrollo de circuitos orientado a estos dispositivos. En esta tesis se señalan algunos puntos débiles de las herramientas de ubicación y cableado actuales, y se desarrollan métodos y estrategias flexibles con el objetivo de hacer frente a las exigencias actuales y futuras en el diseño sobre sistemas reconfigurables. Entre los principales resultados prácticos de la tesis podemos destacar: un método nuevo de Optimización Combinatoria adaptativo basado en principios termodinámicos, una nueva aproximación polinómica al Árbol de Steiner Mínimo Rectilíneo, un método de ubicación y rutado simultáneos sobre FPGAs, y un método de partición sobre mallas de FPGAs basado en ubicación y rutado.

A mis padres, a Nuria y a Pablo

Agradecimientos

En primer lugar quiero agradecer a Juan Lanchares por el magnífico trabajo desarrollado en la dirección de esta tesis. Su confianza, esfuerzo y apoyo me han permitido ir cubriendo distintos objetivos necesarios para la realización de esta investigación.

También quiero dar las gracias a Juan Manuel Sánchez por su dirección durante los dos primeros años del doctorado. Agradecer asimismo a Román Hermida por el interés mostrado en la investigación, y por el acertado trabajo realizado en la corrección tanto de la tesis como de los distintos artículos sobre los que se asienta.

Me gustaría dar las gracias a todos mis compañeros de la ETSIAN. Especialmente a Miguel Ángel Alvaro por su valiosa discusión técnica sobre distintas cuestiones suscitadas a lo largo de este trabajo, a Manuel Laguna, Javier Lozano, Vicente Gallego, Mauricio Bados y Alfredo Vizcaya por su generosa ayuda a lo largo de estos años, y a los sucesivos directores de la escuela por el apoyo recibido.

Agradecer a Francisco Tirado por la confianza mostrada al incluirme en el proyecto de investigación CICYT TIC 99-0474 del cual es investigador principal. A José Ignacio Hidalgo y Oscar Garnica por su ayuda en distintos aspectos de la investigación. A Manuel Ortega, Antonio Vaquero y Milagros Fernández por su orientación inicial, y en general a todos los miembros del Departamento de Arquitectura de Computadores y Automática de los cuales he recibido un gran apoyo. Quiero también dar las gracias a Vaughn Betz y Jonathan Rose de la Universidad de Toronto por facilitarme el conjunto de benchmarks utilizados en esta tesis.

Gracias a mis padres, hermanos y familia política por su cariño y ayuda, los cuales han facilitado enormemente este trabajo. Agradecer a mis amigos por el apoyo recibido, y dedicar un recuerdo especial a Gerardo, que sin duda se habría alegrado en este momento.

Finalmente, quiero agradecer a mi mujer Nuria y a mi hijo Pablo por su amor, paciencia y comprensión, los cuales han sido mi mayor soporte.

Tabla de contenidos

Capítulo 4

Ubicación en FPGAs mediante

Optimización Combinatoria Termodinámica

Capítulo 5

Estimación de la congestión en la fase de ubicación:

Valores Esperados de Ocupación

Capítulo 6

Medida de la congestión en la fase de ubicación:

Regiones de Steiner Rectilíneas

Capítulo 1

Introducción

Las FPGAs *(Field Programmable GateArrays)* [Xil] son dispositivos de lógica programable por el usuario que proporcionan un método rápido y barato para el desarrollo de circuitos. Sus características de flexibilidad y reconfigurabilidad proporcionan interesantes ventajas para el apoyo al diseño y a la producción de sistemas digitales. Las FPGAs fueron introducidas inicialmente, a mediados de los ochenta, como una tecnología de propósito específico. Sin embargo, hoy en día su ámbito de aplicación se extiende a distintas áreas como son la emulación lógica, el prototipado rápido, los sistemas multimodo, las computadoras reconfigurables o problemas concretos de supercomputación.

Para obtener beneficio de las posibilidades que ofrecen los sistemas basados en FPGAs, es necesario disponer de herramientas que automaticen los procesos implicados en el diseño y desarrollo de circuitos orientado a estos dispositivos. Así, se debe facilitar la carga automática de algoritmos sobre las FPGAs, de la misma forma que se cargan programas sobre las computadoras de propósito general. Mientras que la implementación de un algoritmo sobre una computadora de propósito general da como resultado un programa, en el caso de la carga de un algoritmo sobre una FPGA, el resultado es un circuito dedicado. Este circuito dedicado puede satisfacer distintas aspiraciones como son: sustituir el software por hardware para proporcionar mayores prestaciones, emular diseños hardware para depurar prototipos de circuitos integrados, o diseñar sistemas digitales fácilmente actualizables. La carga de un algoritmo sobre una FPGA es un proceso complejo, que normalmente se realiza a través de las distintas etapas correspondientes al diseño de circuitos VLSI (*Very Large Scale Integration*)

Una etapa importante del ciclo de diseño VLSI orientado a FPGAs es el diseño físico. El diseño físico consiste en la transformación de la representación circuital de un

sistema, en una representación geométrica sujeta a las reglas impuestas por la tecnología de fabricación. Las FPGAs son dispositivos formados por un conjunto de elementos lógicos y de interconexión programables. Al ser dispositivos prefabricados, el ciclo diseño físico difiere en algunos aspectos del diseño físico orientado a otras tecnologías. Así, la anchura de los canales de la FPGA, a través de los cuales se realiza el cableado del circuito, constituye una ligadura inflexible que requerirá, en muchas ocasiones, una distribución cuidadosa de las redes que componen el circuito sobre la FPGA. Como las alternativas para realizar esta distribución crecen de forma exponencial con el tamaño del circuito, el ciclo de diseño físico de un sistema basado en FPGAs se convierte en un proceso complejo, incluso para tamaños moderados de circuito. Por este motivo, el diseño físico de un circuito sobre una FPGA normalmente se descompone en diversas etapas como son: la partición, la ubicación y el rutado. La partición divide el circuito lógico en partes, de forma que cada una se pueda implementar mediante la programación de un bloque lógico configurable prefabricado. En la ubicación se realiza la asignación específica de los bloques lógicos a bloques físicos. Y por último, el rutado selecciona los recursos para realizar las conexiones entre los distintos bloques físicos.

En cada una de estas etapas, las herramientas de diseño físico deben ser capaces de administrar correctamente los recursos lógicos y de cableado. Por ejemplo, una correspondencia inadecuada entre bloques lógicos y físicos provoca la congestión de señales, y por lo tanto, el agotamiento de los recursos de cableado. Ello obliga al usuario a recurrir a FPGAs de mayor tamaño y coste, provocando normalmente una infrautilización de los recursos lógicos de los sistemas basados en FPGAs. Por lo tanto, las herramientas de diseño automático se deben encargar de hacer cumplir las restricciones impuestas por los sistemas. Entre estas restricciones destaca, en el caso de las FPGAs, la escasez de líneas de interconexión entre los elementos lógicos que las componen, mientras que en el caso de los sistemas compuestos por varias FPGAs o sistemas Multi-FPGAs, su principal carencia la constituye el reducido número líneas de comunicación disponibles entre FPGAs. En ambos casos, una ubicación descuidada hace imposible el rutado.

Un aspecto importante a considerar en la automatización del diseño físico es la complejidad de tipo NP de los problemas tratados. Como ya comentamos anteriormente, el número de alternativas o configuraciones posibles del circuito, crece de forma exponencial con el tamaño de éste. Ello hace que la búsqueda de las mejores soluciones, o simplemente de soluciones viables, sea computacionalmente cara. El éxito de las herramientas de diseño automático estará determinado fundamentalmente por los algoritmos utilizados para la selección de soluciones, y por los criterios que se establezcan para estimar o evaluar la calidad de éstas.

En cuanto a los algoritmos para la selección de soluciones, la mayoría de las herramientas CAD (*Computer Aided Design*) abordan estos problemas mediante algoritmos de optimización combinatoria como el *Enfriamiento Simulado* o los *Algoritmos Genéticos*. En la actualidad, el *Enfriamiento Simulado* (ES) es uno de los algoritmos más utilizados, debido a que proporciona buenos resultados en tiempos aceptables. Su ámbito de aplicación se extiende también a multitud de problemas derivados de las ciencias e ingenierías. Así, aparece referenciado en áreas tan dispares como las redes neuronales, las finanzas o el análisis de combate. Sin embargo, el ajuste fino de este algoritmo para los distintos problemas y funciones de coste, requiere costosos estudios experimentales para determinar el valor de los parámetros que lo controlan. Debemos tener en cuenta que la resolución de los problemas descritos anteriormente, exige la exploración de múltiples funciones de coste con el objetivo de determinar la más idónea para cada problema. De forma que la tarea asociada al ajuste de los parámetros para cada una de esas funciones de coste, puede llegar a consumir gran parte de los esfuerzos de la investigación. Por lo tanto, la automatización del ajuste de parámetros del método de optimización combinatoria utilizado podría ser muy beneficiosa para el éxito de la investigación.

Por otra parte, podemos establecer diversos criterios para discriminar la calidad de las soluciones exploradas por las herramientas de diseño físico. Estos criterios pueden ir desde estimaciones poco fiables pero rápidas, a medidas más detalladas y lentas. Como consecuencia, la elección de uno u otro criterio, se debe realizar teniendo en cuenta diversos factores como son: la complejidad de la estimación o medida, el tiempo de

computo de ésta, el tamaño de la instancia del problema y el algoritmo de optimización utilizado, la calidad exigida para la solución, la configuración de partida, etc. En el caso de la ubicación, para discriminar entre soluciones en el proceso de búsqueda, es habitual recurrir a una estimación rápida del cable necesario para rutar el circuito. Estas estimaciones permiten abordar el problema de la ubicación de grandes circuitos, pero a costa de producir ubicaciones subóptimas. Aunque sería deseable la medida directa de la calidad de las ubicaciones basada directamente en el cableado, esta medida es excesivamente costosa en tiempo cuando se recurre a las actuales herramientas de cableado. En cuanto a la partición de circuitos orientada a mallas de FPGAs, muchas de las herramientas desarrolladas hasta la fecha no tienen en cuenta el consumo de recursos debido a la conexión entre FPGAs no adyacentes, y por lo tanto, dan lugar a soluciones poco realistas.

De acuerdo con los problemas expuestos, el objetivo de este trabajo de investigación ha ido dirigido a la obtención de métodos eficaces para el diseño físico de circuitos sobre sistemas basados en FPGAs. Para alcanzar ese objetivo es necesario cubrir diversas etapas, tanto en el diseño físico orientado a FPGAs como a sistemas Multi-FPGAs, que resuelvan los problemas planteados. A continuación se resumen los objetivos comunes y específicos para cada uno de estos sistemas:

• Estudio e investigación de **métodos de optimización combinatoria** que, no sólo generen soluciones de calidad, sino que también proporcionen una mayor flexibilidad para facilitar y mejorar el tratamiento de los problemas considerados. En concreto, reducción del esfuerzo y tiempo invertido en el ajuste de los parámetros de los algoritmos del *Enfriamiento Simulado*, y en la medida de lo posible, automatización de este proceso. La automatización del ajuste de parámetros permitiría concentrar el esfuerzo en la investigación de nuevas funciones de coste que contemplen los problemas específicos tanto en FPGAs como en mallas de FPGAs.

• Estudio de los problemas de congestión producidos en el **diseño físico orientado a FPGAs**, y desarrollo de métodos que permitan evitarla en las dos etapas más influyentes

del ciclo de diseño físico: fases de ubicación y cableado. En lo que a la fase de ubicación se refiere, estudio de su influencia en la congestión, determinando las virtudes y carencias de las funciones de coste comúnmente utilizadas. Desarrollo de nuevos métodos y funciones de coste que aborden los problemas de congestión. En cuanto a la fase de cableado, estudio de los algoritmos de cableado actuales determinando las causas de su lentitud, y búsqueda de algoritmos de cableado alternativos que sean rápidos y no comprometan la calidad de las soluciones.

• Análisis del problema del **diseño físico de circuitos orientado a Sistemas Multi-FPGA** con topología de malla. Estudio del consumo de recursos de cableado en la conexión entre FPGAs no adyacentes, y desarrollo de métodos y estrategias dirigidas a mejorar los bajos porcentajes de utilización de la lógica que presentan estos sistemas.

A continuación describimos la estructura de esta tesis que se desarrolla según el siguiente esquema:

En el capítulo 2 se introducen los distintos estilos de diseño de circuitos integrados. Se destacan los sistemas reconfigurables y las diferentes familias de FPGAs. Posteriormente, se realiza una descripción general de los elementos fundamentales que componen tanto las FPGAs como los sistemas Multi-FPGA. Se exponen los problemas derivados del diseño físico sobre FPGAs y sistemas Multi-FPGA, y se analizan las virtudes y defectos de los procedimientos y herramientas desarrollados hasta la fecha.

En el capítulo 3 se presenta el algoritmo de *Optimización Natural (ON)*, un método auto-sintonizable para el tratamiento de problemas de ubicación. Asimismo se proporcionan resultados experimentales comparándolo con el algoritmo de *Enfriamiento Simulado*.

En el capítulo 4, se presenta *OCT* (*Optimización Combinatoria Termodinámica*), un método nuevo de optimización combinatoria basado en principios termodinámicos, que se adapta automáticamente a distintos problemas y funciones de coste. Su efectividad se pone de manifiesto mediante comparaciones con una herramienta que utiliza el algoritmo de *Enfriamiento Simulado*.

En el capítulo 5 se profundiza en el problema de la congestión en FPGAs, y se desarrolla el modelo probabilístico *Valores Esperados de Ocupación (VEO)* para estimar la congestión local en la fase de ubicación. Se muestra la fiabilidad del modelo a través de un conjunto de resultados experimentales y se discute su ámbito de aplicación.

El capítulo 6 enlaza con la problemática del capítulo anterior y justifica un cambio de orientación en la aproximación a la solución del problema de la congestión local. Se presenta el algoritmo de cableado *Regiones de Steiner Rectilíneas (RSR)*, el cual es una aproximación polinómica a la solución del problema del *Árbol de Steiner Mínimo Rectilíneo (ASMR)*, y se compara con las mejores aproximaciones al *ASMR* desarrolladas hasta la fecha. Asimismo, se describe una estrategia de ubicación en etapas *(U-RSR)* facilitada por el algoritmo de *Optimización Combinatoria Termodinámica,* y cuyo objetivo es reducir la congestión local realizando medidas de la misma con el algoritmo *RSR*. Para terminar este capítulo, se presenta un algoritmo de cableado global basado en las regiones *RSR (CG-RSR)*, y se comparan los resultados producidos por la estrategia conjunta de ubicación y cableado (*UCG-RSR)*, con la herramienta *Versatile Place&Route* (VPR) de la Universidad de Toronto.

En el capítulo 7 se analizan las causas de la baja utilización de los recursos lógicos en los sistemas Multi-FPGAs. Se presenta un modelo para describir las mallas de FPGAs, y se realiza la formulación del problema de partición y ubicación dentro de ese modelo. Finalmente, se diseña una estrategia para obtener un buen compromiso *tiempo/calidad* para la resolución del problema basada en los algoritmos de *OCT* y *RSR* aplicándola a un conjunto de circuitos de prueba.

Por último, en el capítulo 8 se realiza una recopilación de las principales aportaciones de la tesis así como las futuras líneas de investigación a seguir.

Capítulo 2

Arquitectura de los sistemas reconfigurables y herramientas de diseño automático

2.1 Los sistemas reconfigurables

Los sistemas reconfigurables están proporcionando una gran flexibilidad para la producción de sistemas digitales. Las posibilidades que ofrecen los sistemas reconfigurables abren nuevos caminos como el prototipado rápido, la emulación lógica, las computadoras reconfigurables o el hardware multimodo. Podemos encontrar distintos tipos de circuitos reconfigurables [BR96] como son los SPLDs (*Simple Programmable Logic Devices*), los CPLDs (*Complex Programmable Logic Devices*) o las FPGAs (*Field Programmable Gate Arrays*). Los primeros en aparecer fueron los SPLD, que permiten implementar funciones lógicas en forma de suma de productos. Conforme la tecnología avanzó, se hizo posible la integración de varios SPLDs en un único chip constituyendo los CPLDs. Frente a estos dispositivos programables existían otros dispositivos lógicos de propósito general con una mayor capacidad de integración como son los arrays de puertas o MPGAs (*Mask-Programmable Gate Arrays*). Los MPGAs están formados por una matriz de transistores prefabricados cuyas conexiones se pueden configurar durante el proceso de fabricación del chip, dando lugar a un circuito específico. Con este panorama, aparecieron las FPGAs que proporcionaban un punto de encuentro entre ambas tecnologías. Así, las FPGAs surgieron originalmente con una doble identidad. Por una parte, se podían considerar MPGAs baratos, lentos y de baja densidad lógica. Por otra parte, eran como PLDs de elevado coste y mayor capacidad lógica. Las FPGAs son, por lo tanto, circuitos integrados que disponen de un conjunto de recursos lógicos y de interconexión prefabricados. El usuario sólo tiene que programarlos para darles una determinada funcionalidad. De esta forma, las FPGAs

proporcionan dispositivos digitales de alta escala de integración configurables por el usuario. Tanto el tiempo como los costes de desarrollo se ven reducidos drásticamente. Existen básicamente dos tecnologías para la programación de las FPGAs: los antifusibles y las celdas de memoria SRAM. Las FPGAs de antifusibles no son reprogramables ni volátiles. Por el contrario, las FPGAs programables por celdas de memoria SRAM son volátiles y reprogramables. De aquí en adelante nos centraremos en estas últimas por ser las que mayor interés despiertan. Aunque la volatilidad de las SRAM-FPGAs fue considerada inicialmente como un gran inconveniente, con el tiempo ese defecto se ha convertido en su principal virtud. Puesto que son dispositivos reprogramables, su configuración se puede cambiar fácilmente para depurar diseños o poner al día los sistemas. Entre las aplicaciones típicas de las FPGAs se pueden citar: la lógica aleatoria, la integración de varios SPLDs en uno sólo, los dispositivos de control, los dispositivos de codificación y de filtrado en comunicaciones, y en general cualquier sistema de tamaño pequeño o mediano. Existe otra serie de aplicaciones que requieren mayor capacidad lógica, y por lo tanto, exigen la interconexión de varias FPGAs formando sistemas Multi-FPGAs. Entre estas aplicaciones están la generación de prototipos correspondientes a diseños sobre arrays de puertas, o la emulación y depuración de grandes sistemas hardware. De la misma forma, los sistemas basados en FPGAs se pueden utilizar con diferentes configuraciones en sistemas multimodo o computadoras reconfigurables. Otra aplicación emergente basada en FPGAs es el entrenamiento y ejecución de redes neuronales. A continuación profundizamos en estas y otras posibilidades que ofrecen las FPGAs en el diseño de sistemas digitales.

La emulación lógica de circuitos de aplicación específica (ASICs) [VBB93, DBTHHA94, WSKR99] es una de las aplicaciones de mayor éxito de los sistemas reconfigurables. Los diseños de ASICs deben ser verificados para asegurar que el circuito realiza sus funciones correctamente. Esta tarea tradicionalmente se ha realizado por simulación software de forma lenta. En emulación lógica, el circuito se implementa sobre un sistema formado por varias FPGAs (sistema Multi-FPGA). De esta forma, se puede acelerar la verificación del ASIC en varios ordenes de magnitud. Por otra parte, distintos procesadores [Gat95, Gan96] y arquitecturas [Mei95] también se están

emulando en sistemas Multi-FPGAs dada la versatilidad que proporcionan estos dispositivos.

De forma similar, los sistemas reconfigurables facilitan el desarrollo rápido de prototipos [BEKH94, KG95, MK98, KRWDP99], los cuales permiten, en algunos casos, realizar pruebas de integración de los componentes digitales a nivel de sistema incluso antes de su fabricación como circuito integrado [RBB91].

Otra área con gran futuro para las FPGAs son las computadoras reconfigurables [HWGG93, HKH94, Bax99, BR2000, SLLKBFM2000]. Estas computadoras estarán constituidas por procesadores y lógica programable de forma que podrán proporcionar hardware especializado para la aplicación que esté siendo ejecutada. Los beneficios que se pueden obtener son similares a los que ofrecen los procesadores con conjuntos de instrucciones diseñados para aplicaciones específicas (ASIP). Este tipo de procesadores permite acelerar la ejecución de las aplicaciones para los que están diseñados. Las computadoras reconfigurables obtendrán ese mismo beneficio sin necesidad de comprometer el hardware, de forma que se podrán adaptar a distintas aplicaciones.

Igualmente, el uso de sistemas reconfigurables hace que la supercomputación tenga un menor coste [Wau91, YNK96], y por tanto sea más accesible para todos los usuarios. La flexibilidad de las FPGAs permite explotar el paralelismo a niveles de bit, aritmético, de instrucción o de aplicación [CEA95, CW98, HBS98, WK99]. El potencial de las FPGAs en este campo ya se ha hecho evidente con el desarrollo de diferentes procesadores para investigación. Así, el sistema Splash [Gok90] configurado para la resolución de problemas de correspondencia de cadenas genéticas, mejoró considerablemente el rendimiento de otras implementaciones basadas en supercomputadores. El sistema DECPeTLe-1, también es citado por sus aplicaciones y récords en distintas disciplinas como la criptografía [VBRSTB95], o la física de altas energías [MVB95]. Por otra parte, se diseñan aceleradores hardware para aplicaciones concretas, como es la simulación de la dinámica de n cuerpos [CKL95], o coprocesadores para criptografía [PT98] y procesamiento vectorial [AM98]. Asimismo, se han realizado distintas implementaciones basadas en FPGAs de algoritmos genéticos para la resolución de problemas de optimización [GN96, BM98, KBHBKA98, SMP99, HZ2000]. También,

se han propuesto otros modelos basados en FPGAs para reducir los tiempos de computo en la resolución de problemas diversos como el cálculo de órbitas de satélites [BS98], o problemas de física estadística [MI93, MC93].

Los sistemas reconfigurables también presentan un futuro interesante en sectores como el de las comunicaciones [CME93, MGTG97, TBD98], los dispositivos multimedia [EWJ95, MO98] o sistemas de vídeo [LADDRSS98, PWH99, MKBGHS2000]. Por una parte, las FPGAs permiten salvar las restricciones de área que imponen algunas de estas aplicaciones, posibilitando el diseño de interfaces reconfigurables más versátiles [RPM91, SRR98]. Por otra parte, el paralelismo potencial de las FPGAs puede proporcionar el elevado rendimiento computacional necesario para el procesamiento digital de las señales (DSP) requerido en muchas de estas aplicaciones [BM94, Sch95, KVGO99]. Además, las FPGAs permiten una rápida actualización de los productos, la cual es imprescindible en estas áreas debido a sus continuos avances. También se empiezan a describir aplicaciones DSP con FPGAs en otras áreas de conocimiento como la visión artificial [KDCA96, WH97, AP99], el reconocimiento del habla [YSS97], el radar [MM99] o el sonar [GN98].

Otro de los campos donde las FPGAs despiertan especial interés es el de las redes neuronales [DHY93, BH94, BT95, PDBS98]. Las redes neuronales son modelos computacionales basados en la estructura de las neuronas del cerebro. Uno de los aspectos más importantes de estos modelos es que los elementos básicos que componen la red se configuran para cada problema mediante un proceso de *aprendizaje*. Las posibilidades de reconfiguración que ofrecen las FPGAs pueden facilitar este proceso de aprendizaje debido que la configuración hardware se puede modificar en todo momento.

En este mismo sentido, aparecen nuevas técnicas agrupadas bajo el nombre de *reconfiguración dinámica* [Tud95, JTYCS98, MMFC98]. Entre otras posibilidades, la reconfiguración en tiempo de ejecución permitiría disponer de un *hardware virtual* equivalente a la *memoria virtual* disponible en las computadoras actuales [LD93, LS96, Wer97, BDHSW97, MDGS98, ML99]. Es decir, la reconfiguración permite ahorrar hardware reutilizando el mismo recurso con distintas configuraciones.

Hasta aquí, se ha mostrado una revisión de algunas de las aplicaciones actuales de los sistemas reconfigurables. También se han mencionado algunas líneas de investigación que previsiblemente orientarán las futuras aplicaciones de los sistemas basados en FPGAs. Sin embargo, para hacer realidad todo este potencial que presentan las FPGAs, es necesario el desarrollo de herramientas automáticas que permitan el tratamiento, de forma eficiente, de la complejidad creciente de los circuitos. Como veremos a continuación, de acuerdo con el actual ciclo de diseño VLSI, es en la etapa de diseño físico cuando se realiza la correspondencia entre la representación del circuito y el dispositivo físico final que lo soporta. En el caso de las FPGAs, este proceso se realiza habitualmente descomponiendo el circuito en subcircuitos o bloques, de forma que la funcionalidad de cada bloque se corresponda con la que se puede proporcionar mediante la configuración de un bloque de lógica prefabricado. En una segunda fase, se realiza la ubicación o correspondencia entre bloques lógicos y físicos de acuerdo con algún criterio de proximidad. Finalmente, la comunicación entre subcircuitos relacionados, se establece configurando los recursos de cableado programables distribuidos a lo largo y ancho de toda la FPGA. La programación de una FPGA lleva asociadas, por lo tanto, una serie de tareas complejas cuya realización óptima requiere la resolución de distintos problemas de optimización combinatoria, los cuales resultan inabordables sin la ayuda de computadoras. De esta forma, la automatización se convierte no sólo en un apoyo, sino en un objetivo irrenunciable.

En las siguientes secciones se introducen las ideas básicas relativas al diseño *VLSI (Very Large Scale Integration),* y a las distintas metodologías o estilos de fabricación. Seguidamente, se repasa el estado de la tecnología de las FPGAs a través de las diferentes familias de dispositivos que proporcionan los distintos fabricantes. Asimismo, se realiza una revisión general de los elementos que componen las FPGAs centrándonos, de manera más detallada, en aquellos elementos que son relevantes para la descripción de los problemas planteados en este trabajo de investigación. Ese mismo tratamiento se dará a los sistemas Multi-FPGA. Finalmente, se exponen los problemas derivados del diseño físico orientado a FPGAs y sistemas Multi-FPGA, y se analizan las

virtudes y defectos de los procedimientos y herramientas desarrollados hasta la fecha para su resolución.

2.2 Ciclo de diseño VLSI

El ciclo de diseño de circuitos de muy alta escala de integración o *VLSI*, comprende una serie de etapas que van desde la especificación de los requisitos que debe cumplir el circuito integrado al producto final, es decir, al chip empaquetado. Entre estas dos fases podemos distinguir una serie de pasos a los que se les denomina *ciclo de diseño VLSI*. De acuerdo con [She99], este ciclo se compone de las siguientes etapas:

1. Especificación del sistema: La especificación del sistema normalmente recoge un compromiso entre los requisitos del mercado, la tecnología y la viabilidad económica del producto. Las especificaciones deben incluir la funcionalidad, la velocidad, la potencia y las dimensiones físicas del chip.

2. Diseño de la arquitectura: A partir de las especificaciones, en esta etapa se diseña la arquitectura del sistema. Este diseño incluye decisiones relativas a las instrucciones que ejecutará el sistema, al grado de paralelismo espacial y temporal, a la jerarquía de memoria, etc. El resultado de este diseño es una descripción en forma de texto denominada *Especificación de la Micro-Arquitectura* o *MAS (Micro-Architectural Specification)*. En este documento se debe incluir una predicción de la velocidad, potencia y tamaño del circuito integrado.

3. Diseño funcional o de comportamiento: En esta etapa se identifican las unidades funcionales del sistema y sus requisitos de interconexión. Se describe el comportamiento de cada unidad funcional mediante sus entradas, salidas y tiempo, sin especificar la estructura interna. El resultado del diseño funcional es una descripción que relaciona las distintas unidades funcionales, y permite la emulación del sistema entero para su depuración.

4. Diseño lógico: En la etapa de diseño lógico se derivan, a partir de la descripción de comportamiento, las operaciones aritméticas y lógicas, el flujo de control, el tamaño de las palabras y la asignación de registros. El resultado se denomina descripción a nivel de transferencia de registros (*RTL-Register Transfer Level*), y se expresa habitualmente en un lenguaje de descripción de hardware (*HDL- Hardware Description Language*) como *VHDL o Verilog*. La descripción *RTL* está formada por expresiones booleanas e información de tiempo, y se puede utilizar para simulación y verificación. En esta etapa, las expresiones booleanas se optimizan de acuerdo con los objetivos de área, tiempo, potencia, etc. En ocasiones, el paso de la descripción funcional a *RTL* se puede automatizar por medio de herramientas de síntesis de alto nivel (*High Level Synthesis*).

5. Diseño del circuito: Durante esta etapa, las expresiones booleanas se convierten en una representación del circuito, teniendo en cuenta los requisitos de potencia y velocidad. La representación del circuito se le suele denominar lista de redes *(netlist*), y contiene los distintos elementos (macros, puertas, etc.) y sus interconexiones. Mediante la simulación del circuito en este nivel se pueden verificar su correcto funcionamiento. Las herramientas de síntesis lógica permiten generar automáticamente la *netlist* a partir de la descripción *RTL*.

6. Diseño físico: En la fase de diseño físico, la representación circuital o *netlist*, se transforma en una representación geométrica del circuito denominada composición (*layout*). Durante la fase de diseño físico, los componentes lógicos se transforman en *patrones geométricos*, de acuerdo con las reglas de diseño dictadas por la tecnología y el proceso de fabricación. Las interconexiones entre componentes normalmente se realizan, utilizando diversas capas, a través de regiones dedicadas. En esta etapa, también se realizan distintas verificaciones del diseño.

7. Fabricación: Una vez verificado el diseño, los datos de la composición se mandan a fabricar. En el proceso de fabricación, los datos correspondientes a las distintas capas de la composición se transforman en máscaras fotolitográficas. El proceso de fabricación

se realiza en varias etapas durante las cuales se depositan y difunden, a través de las máscaras, diversos materiales sobre la oblea de silicio. Dependiendo del tamaño del circuito integrado, con una oblea de 20cm o 30cm de diámetro se pueden fabricar varios chips.

8. Empaquetado, prueba y depuración: En la última etapa la oblea se trocea en chips individuales. Los chips utilizados en placas de circuitos impresos *(PCB-Printed Circuit Board)* se empaquetan y se prueban para asegurar que cumplen las especificaciones, mientras que los chips diseñados para Módulos Multi-Chip *(MCM- Multi-Chip Modules)* no se empaquetan al ser integrados directamente en una plataforma mayor.

2.3 Estilos de diseño físico

El ciclo de diseño físico [She99] es un proceso complejo por lo que normalmente se divide en varias etapas. Para manejar la dificultad del diseño físico han surgido unos pocos *estilos* que responden a distintas filosofías de diseño, costes y tiempos de puesta en el mercado (*time-to-market*). Entre esos estilos podemos citar: *FC (Full-Custom), SC (Standard Cell), GA (Gate Arrays), FPGA (Field Programmable Gate Arrays) y SG (Sea of Gates).*

El estilo *FC* permite mezclar patrones geométricos correspondientes a bloques funcionales de distintos tamaños. La partición del circuito en bloques funcionales se realiza habitualmente de forma jerárquica. Debido a su complejidad, se suele utilizar en la fabricación de microprocesadores donde prima la calidad sobre tiempo de diseño.

En el estilo *SC* se recurre a una biblioteca de celdas standard (500-2000 unidades) para realizar la composición, donde cada celda está caracterizada por su funcionalidad y rendimiento. El circuito se divide en bloques de forma que cada uno sea equivalente a una celda predefinida. Las celdas son rectangulares, de la misma altura, y se disponen en filas dejando canales intermedios entre las filas para el cableado. La complejidad del diseño físico se reduce respecto al estilo *FC*. El estilo *SC* es inherentemente no jerárquico por lo que se utiliza principalmente en lógica aleatoria y de control. Salvada

la inversión inicial en el desarrollo de la biblioteca de celdas, el diseño en *SC* es más rápido que en *FC* sin alcanzar el rendimiento ofrecido por este último. El estilo *SC* se ajusta mejor a circuitos de tamaño moderado y volúmenes de producción medios.

En el estilo *GA* existe un array de celdas prefabricadas de igual tamaño (puertas lógicas), separadas por canales horizontales y verticales para el cableado. El circuito se modifica de forma que se pueda dividir en bloques idénticos, en un número menor o igual que el número de puertas disponibles. La ventaja de este estilo de diseño es que sólo las últimas etapas del proceso de fabricación dependen del circuito diseñado. Por esta razón, la fabricación de un circuito con esta tecnología es más barata.

De forma similar a los *GA*, las FPGAs están formadas por un array de celdas prefabricadas separadas por canales horizontales y verticales para el cableado. La ventaja de las *FPGAs* sobre los *GA* es que esas celdas, normalmente de mayor complejidad, son programables por el usuario. Además, la mayoría de las *FPGAs* no sólo son programables, sino que se pueden reprogramar con las innumerables ventajas que esto conlleva. Como contrapartida frente a los GA, el nivel de integración en las *FPGAs* es menor, y los tiempos de retardo mayores.

Por último, el estilo SG mejora a los GA en el nivel de integración debido a que todo el área del circuito integrado está llena de transistores. El cableado se realiza utilizando puertas lógicas con lo que se pierde efectividad en área, o añadiendo más capas para conexiones en el circuito integrado, lo que encarece el tiempo y coste de fabricación. En el resto del capítulo nos centraremos en el estilo de diseño de FPGAs, el cual es el objeto de esta tesis.

2.4 Familias de FPGAs

Las *FPGAs* son dispositivos de lógica programable inicialmente orientados a *Circuitos de Aplicación Específica (ASICs- Application Specific Circuits),* debido a que permiten reducir drásticamente el tiempo de puesta en el mercado de estos dispositivos.

En la actualidad, la capacidad de las FPGAs está por encima del millón de puertas lógicas lo que permite pensar en el desarrollo de sistemas digitales enteros integrados en estos dispositivos programables. Existen diferentes familias de FPGAs reconfigurables

como son las Xilinx [Xil], Altera[Alt], Lucent Orca [Luc] y Actel [Act]. Una característica común a casi todas estas familias, es el uso de LUT#s (Look-Up Tables) para la generación de cualquier función lógica de # entradas. El número de entradas por LUT puede variar de unas FPGAs a otras, siendo 4 o 5 valores habituales. En cuanto a la estructura de los bloques lógicos o celdas básicas, sí se encuentran diferencias significativas de unas familias a otras. En la Tabla 2.1 se reproduce un estudio reciente [KS2000] donde se muestra las características principales de estas familias como son: capacidad del mayor dispositivo disponible de la familia, tipo y número de bloques lógicos o celdas básicas, número de pines de entrada/salida y tecnología de fabricación. En la Tabla 2.2 se muestra una comparación de la capacidad de estos dispositivos medida en puertas lógicas.

Tabla 2.1 *Familias de FPGAs*

Familia	Dispositivo mayor	Bloque Lógico	Capacidad Celdas básicas	Entrada/ Salida	Tecnología
Xilinx 4000	XC40250XV	CLB	8,464 CLBs	448	SRAM (0.25µ)
Xilinx Virtex, Virtex-E	XVC3200E	CLB	16,000 CLBs	804	SRAM (0.18µ)
Altera FLEX 8000	81500	LE	1,296 LEs	208	SRAM (0.5µ)
Altera FLEX 10k	EPF10K250	LE	12,160 LEs	470	SRAM (0.25µ)
Altera Apex20k	EP20K1500K	LE	51,840 LEs	808	SRAM (0.18µ)
Actel ProAsic	A500K510	tile	51,200 tiles	623	Flash (0.25µ)
Lucent ORCA	OR3L225B	LUT	11,552 LUTs	612	SRAM (0.25µ)

Tabla 2.2 *Capacidad lógica de las distintas familias de FPGAs*

Familia	Dispositivo mayor	Celdas lógicas (LC) por Bloque lógico (1)	Capacidad Puertas eq. (lógica y RAM)	Capacidad Puertas (sin memoria)
Xilinx 4000	XC40250XV	2.375	250,000	241,000
Xilinx Virtex-E	XVC3200E	4.5	4,074,387	876,096
Altera FLEX 800	81500	1	16,000	16,000
Altera FLEX 10k	EPF10K250	1	250,000	149,000
Altera Apex20k	EP20K1500K	1	1,500,000	622,000
Actel ProAsic	A500K510	0.4	410,000	240,000
Lucent ORCA	OR3L225B	1	340,000	166,000

(1) Celda lógica(LC) ~ LUT4+FF (Definición introducida por Xilinx [Xil] *para la comparación entre distintas tecnologías)*

Analizando las tablas 2.1 y 2.2, se observa que el número de celdas básicas varía con la granularidad. La celda básica en Xilinx es el CLB. Sin embargo, la estructura de un CLB es diferente en las familias XC4000 y Virtex. Así, un CLB de la familia XC4000 contiene 2 LUT4 (Look-up table), 1 LUT3 y 2 FFs, mientras que un CLB de la familia Virtex está compuesto de dos partes, donde cada una de ellas contiene 2 LUT4s, 2 FFs (*Flip-Flops*) y multiplexores. En Altera, el bloque básico es el LE (Logic Element), que contiene 1 LUT4 y 1 FF. Las FPGAs Actel ProAsic tienen una granularidad más fina con celdas denominadas *tile*, las cuales se pueden configurar como funciones de tres entradas o como FFs. Por último, las FPGAs Lucent Orca están basadas en LUT4s. Para hacer posible la comparación entre las distintas tecnologías, Xilinx introdujo la noción de Celda Lógica (LC), la cual consiste en 1LUT4+1FF (equivalente a 12 puertas). De esta forma, el CLB de la familia XC4000 es 2.375 LCs y el de la familia Virtex es de 4.5 LCs. El *tile* de la familia Actel equivale a 0.4 LCs. En la Tabla 2.2 también se muestra una estimación de puertas equivalentes determinada a partir de los datos proporcionados por los fabricantes.

2.5 Arquitectura de las FPGAs

En la sección anterior, se ha realizado una revisión de las distintas FPGAs disponibles en el mercado. Se destacó asimismo la estructura de las celdas básicas como elemento diferencial entre todas ellas. Existen también diferencias en cuanto a la disposición de estas celdas en la FPGA. Además, las FPGAs de última generación empiezan a incorporar nuevos elementos como interfaces para la conexión de memorias externas (*megabytes)*, que sumados a nuevos bloques de memoria (*kilobytes*), y a las tradicionales unidades de memoria distribuidas a lo largo y ancho de toda la matriz de celdas básicas (*bytes*), permiten hablar de organizaciones jerárquicas de memoria. Además, aparecen nuevos elementos para el tratamiento aritmético y la sincronización interna/externa del tiempo. A pesar de las particularidades que ofrecen las distintas familias, también existe una característica común a casi todas ellas como es la organización básica de celdas en filas y columnas. A partir de esa disposición de los bloques lógicos, los recursos programables de propósito general para la interconexión

entre celdas, se disponen normalmente formando canales horizontales y verticales. Al no existir una única arquitectura de FPGAs, las herramientas comerciales para el diseño físico sobre estos dispositivos, tienen en consideración las características particulares de cada familia. Con el objetivo de hacer posible la comparación directa entre herramientas, es habitual en la literatura la presentación de resultados del diseño físico realizado sobre una arquitectura de tipo *array de celdas* [BRV92, LB93, CLWL95, AR96, LW97, WM97, BR97, ACGR98], la cual integra aquellas características que son comunes a la mayoría de las FPGAs. A continuación pasamos a describir los elementos básicos que componen esta arquitectura.

La estructura básica de una FPGA como la de la Figura 2.1 está constituida por una matriz de bloques lógicos configurables (CLBs), cuyos elementos están separados por segmentos de canal horizontales y verticales. En la periferia de la matriz se dispone un conjunto de bloques programables de entrada/salida (*IOBs-Input/Output Blocks*) para la comunicación con el exterior.

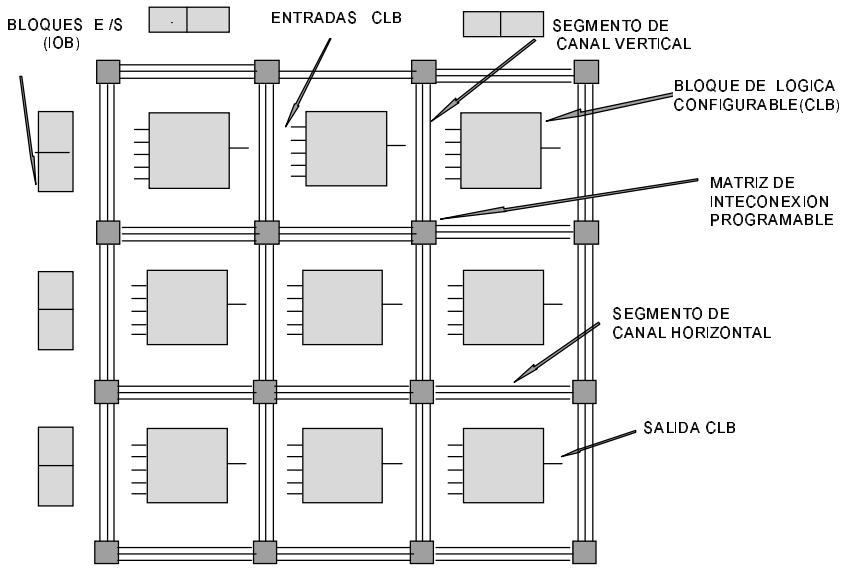


Figura 2.1 *Elementos básicos que componen una FPGA*

Los CLBs son la unidad lógica básica de esta FPGA. Cada CLB se compone de una serie de entradas, salidas y lógica combinacional programable, que permite la

configuración de cualquier función lógica de sus entradas. Asimismo, dispone de elementos de almacenamiento para la implementación de circuitos secuenciales. La estructura interna de los CLBs depende del tipo de FPGA, proporcionando más o menos funcionalidad según sus componentes.

Por otra parte, las FPGAs proporcionan recursos de cableado para la interconexión de las redes que componen el circuito. Entre los canales que forman los elementos de la matriz de CLBs se encuentran los recursos de interconexión de propósito general. Éstos se componen de segmentos de canal y elementos de interconexión programables. Así, cada segmento de canal está compuesto por un conjunto de cables, los cuales desembocan en matrices de interconexión programables situadas en la intersección de los canales horizontales y verticales. Además, tanto a la salida como a la entrada de los CLBs se dispone de dispositivos programables para su interconexión con los cables del segmento de canal más cercano. La configuración de todos estos elementos programables permite el establecimiento de conexiones entre determinados CLBs. Otros recursos de cableado consisten en conexiones directas entre CLBs vecinos, y líneas largas normalmente utilizadas para la interconexión de las redes globales.

Por último, los IOBs proporcionan la frontera entre la lógica interna de la FPGA y los terminales externos del circuito integrado. Los IOBs pueden ser configurados como terminales de entrada o de salida de acuerdo con las necesidades del circuito que se está implementando.

.

2.6 Diseño físico sobre FPGAs

Una parte importante en el diseño de circuitos orientado a FPGAs es el diseño físico. Como ya vimos, el diseño físico [She99] es una etapa del ciclo de diseño de circuitos VLSI [Mich94]. En esta etapa, la representación circuital de los componentes del sistema se transforma en una representación geométrica. En el caso de las FPGAs, este ciclo consiste en la implementación de la lista de redes del circuito mediante los elementos programables, tanto lógicos como de interconexión, de la FPGA. Este proceso se lleva a cabo habitualmente en tres etapas: partición, ubicación y cableado.

Mediante la partición (Figura 2.2), el circuito se divide en bloques de forma que la funcionalidad de cada bloque se pueda implementar con un único CLB, que es la unidad lógica básica de las FPGAs [HOIW94][Sas93]. El problema de la partición orientada a FPGAs [MBS95], es sustancialmente distinto del problema de partición dirigida a otros estilos de diseño. En el caso de las FPGAs, el problema de la partición depende fundamentalmente de la arquitectura de los CLBs con los cuales se va a implementar la lógica del circuito. Por ejemplo, la partición orientada a CLBs que contienen LUTs debe realizarse de forma que el resultado sea un conjunto de funciones lógicas, de cuatro o cinco entradas, sintetizables mediante estos dispositivos.

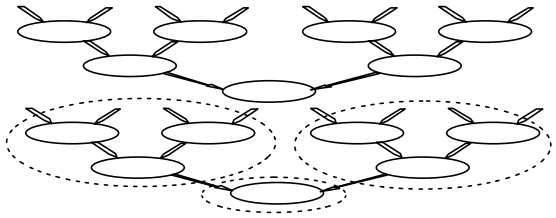


Figura 2.2 *Partición del circuito en CLBs lógicos*

La segunda etapa del ciclo de diseño físico orientado a FPGAs es la ubicación. En la fase de ubicación, los bloques lógicos (CLBs lógicos) obtenidos mediante la partición del circuito se asignan a localizaciones físicas concretas (CLBs físicos) sobre la matriz de CLBs de la FPGA (Figura 2.3).

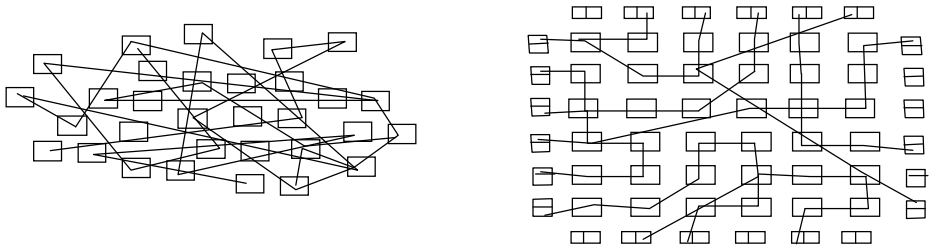


Figura 2.3 *Ubicación de CLBs lógicos en CLBs físicos*

Una vez optimizada la ubicación la siguiente etapa es el cableado, el cual consiste en la conexión de los CLBs físicos relacionados (redes del circuito), mediante la programación de los recursos de interconexión de la FPGA. A continuación pasamos a describir más detalladamente los procesos de ubicación y rutado objeto de este trabajo.

2.6.1 Ubicación sobre FPGAs

La etapa de ubicación es un paso clave en el ciclo de diseño físico al afectar al área y a la velocidad de los circuitos. El problema de la ubicación sobre FPGAs, aún siendo similar a la ubicación sobre arrays de puertas y celdas standard, presenta algunas peculiaridades. En los arrays de puertas y celdas standard, el retardo de las señales debido al cableado proviene únicamente de la distancia que tiene que viajar la señal entre los nodos que se interconectan. Sin embargo, el cableado del circuito sobre las FPGAs se realiza por medio de la interconexión de segmentos de cable a través de interruptores programables [RGS93], los cuales provocan en las señales un retardo mayor que el que se produce en los arrays de puertas y celdas standard. Por lo tanto, el retardo producido en las señales por el cableado sobre FPGAs, es más sensible a la ubicación que en otros dispositivos. Por otra parte, el número de cables por segmento de canal en una FPGA es fijo, de forma que aquellos circuitos que durante el cableado no se ajusten a la capacidad limitada de los segmentos de canal, no se podrán implementar en esa FPGA.

El problema de la ubicación sobre una FPGA se puede especificar como: dado un circuito compuesto de un número fijo de bloques de lógica, bloques de entrada/salida, y de la lista de interconexiones entre estos bloques, asignar posiciones para cada bloque sobre la matriz de CLBs e IOBs de la FPGA, de forma que todas las redes se puedan rutar con los recursos de cableado disponibles. La fase de ubicación tiene una gran influencia sobre la cantidad de cable o recursos de interconexión necesarios para rutar el circuito. Como consecuencia, optimizar la ubicación para el ahorro de cable debe ser un objetivo primordial en el diseño físico. Otro objetivo que frecuentemente se plantea en la fase de ubicación es la minimización del retardo total del circuito. La consecución de

cualquiera de estos objetivos exige realizar una selección de soluciones entre todas las ubicaciones posibles. De manera que, por una parte tenemos que cuantificar los objetivos mediante una función de coste que permita la discriminación entre las distintas soluciones. Por otra parte, y dado que el número de soluciones crece considerablemente con el tamaño del problema, una exploración adecuada del espacio de soluciones exige el uso de algoritmos de optimización combinatoria. En las siguientes subsecciones, revisamos las principales funciones de coste y algoritmos que constituyen el estado del arte de la ubicación. Asimismo, señalamos algunos puntos débiles de estas funciones y algoritmos susceptibles de mejora.

Funciones de coste

Como ya se ha comentado, el principal objetivo de la fase de ubicación es proporcionar una colocación de los bloques lógicos de forma que se facilite el cableado. Por lo tanto, la función de coste en esta etapa debe predecir, para una ubicación dada, la cantidad de cable que será necesaria para rutar el circuito. De forma general, esa cantidad de cable dependerá de la herramienta de cableado utilizada. Sin embargo, sí es posible hacer una primera estimación suponiendo que el cableado de todas las redes se realizará por el camino más corto, lo cual es bastante razonable. El cálculo de las longitudes de una red de dos terminales normalmente se realiza utilizando la distancia rectilínea o de Manhattan. La distancia Manhattan entre dos puntos de coordenadas (x1,y1), (x2,y2) se define como:

$$L=|x1-x2|+|y1-y2| \tag{2.1}$$

Así, podemos descomponer las redes multiterminal en redes de dos terminales, y medir la distancia entre todos los pares fuente/destino (L_i), obteniendo de esta forma la longitud de cable total del circuito. La Ecuación 2.2 muestra la función de coste *longitud de cable total*.

$$\cos te = \sum_{i \in redes} L_i \tag{2.2}$$

En algunos métodos de ubicación analíticos se han utilizado funciones de coste cuadráticas [Hall70, AJK82] de la forma dada por la Ecuación 2.3.

$$Coste = \sum_{i \in redes} L_i^2 \tag{2.3}$$

Sin embargo, se ha observado experimentalmente que esta función mide de forma menos precisa la futura demanda de cable que la función lineal [SDJ91]. Por otra parte, aunque la función lineal dada por la Ecuación 2.2 mide la longitud de cable requerido para redes de dos terminales, ésta pierde precisión según aumenta el número de nodos en redes multiterminal. Como se verá en el capítulo 5, la métrica denominada del *semiperímetro* constituye una estimación mejor de la futura demanda de cable correspondiente a una ubicación dada. El *semiperímetro* estima la longitud de cable necesario para rutar una red multiterminal por la mitad del perímetro del área que la engloba. Por lo tanto, aplicado a todas las redes (Ecuación 2.4), el semiperímetro es una medida del cable total necesario para rutar el circuito. Su minimización permite reducir tanto la congestión global como los retardos de propagación producidos en las señales. Se debe resaltar que ahora, el conjunto de *redes* de la Ecuación 2.4, se refiere al conjunto de redes multiterminal sin descomponerlas en redes de dos terminales.

$$Coste = \sum_{i \in redes} semiperimetro_i \tag{2.4}$$

El semiperímetro también se denomina a veces *Bounding-Box*. Aquí, con el objeto de seguir la notación utilizada en [BR97], denominaremos *Bounding-Box* a la función de coste dada por la Ecuación 2.5, donde se realizan correcciones a la subestima de área de las redes multiterminal calculada con el *semiperímetro*.

$$Coste = \sum_{n=1}^{N_{nets}} \frac{q(n)}{C_{av}} \left[bb_x(n) + bb_y(n) \right] \tag{2.5}$$

Los términos bb_x y bb_y de la Ecuación 2.5 denotan respectivamente los límites horizontal y vertical de la caja que envuelve la red n, q(n) es un factor que compensa la subestima del área de las redes multiterminal [Che94], y C_{av} es la capacidad promedio de un canal en número de pistas. La función de coste *Bounding-Box* será utilizada posteriormente en esta tesis para realizar comparaciones entre los algoritmos de optimización combinatoria, ya que proporciona una estimación rápida y aceptable de la demanda de cable total.

Como hemos señalado anteriormente, la minimización del cable total necesario para rutar el circuito, consigue reducir la congestión global y los retardos en las señales de forma general, pues reduce el camino que tienen que recorrer las señales. Sin embargo, las métricas basadas en la longitud de cable, también pueden generar ubicaciones de los bloques lógicos que provoquen una demanda intensiva de los recursos centrales de interconexión durante el cableado, produciendo congestión en estas zonas [Cro95]. Es necesario, por lo tanto, la investigación sobre nuevas métricas que durante la fase de ubicación realicen estimaciones, y equilibren, las necesidades de cable por segmento de canal [EMHB95]. Se han publicado algunas funciones de coste guiadas por este objetivo [CP86]. En [CP86] se penaliza las ubicaciones en las que se prevé que se va a producir congestión durante el cableado mediante la siguiente función de coste:

$$\sigma = \frac{1}{2}\sum_{i \in redes} longitud(i) + \sum_{i \in Canal\ H} h_i'^2 + \sum_{i \in Canal\ V} v_i'^2 \qquad (2.6)$$

donde

$$h_i' = \begin{cases} h_i - h_{avg} - h_{sd} & si \quad h_i > h_{avg} - h_{sd} \\ 0 & en\ otros\ casos \end{cases}$$

$$v_i' = \begin{cases} v_i - v_{avg} - v_{sd} & si \quad v_i > v_{avg} - v_{sd} \\ 0 & en\ otros\ casos \end{cases}$$

y donde h_i (v_i) es el número de redes que cortan el canal horizontal (vertical) i, h_{avg} (v_{avg}) es la media de h_i (v_i), y h_{sd} (v_{sd}) es la desviación standard de h_i (v_i)

Analizando la expresión, observamos que esta función de coste penaliza el exceso de cortes producidos en un segmento de canal determinado, bien horizontal o vertical. Sin embargo, la penalización se realiza a nivel de canal lo cual asegura que ningún canal soporte un exceso de conexiones, pero no asegura lo mismo para los segmentos de canal.

Otra forma más directa de afrontar el problema de la congestión consiste en una ubicación y rutado simultáneos. Con este método, en cada ubicación se puede medir la ocupación producida en cada segmento de canal mediante un cableado rápido, evitando de esta forma la congestión. Un ejemplo de esta orientación aparece en [NR98], donde se realiza una ubicación y rutados simultáneos. En cada ubicación explorada con un algoritmo de mejora iterativa, se actualiza el cableado de las redes afectadas. Sin embargo, la lentitud de las herramientas de cableado desarrolladas hasta la fecha hace poco viable esta aproximación. Como el autor de este artículo lamenta, la falta de algoritmos de cableado constructivos, le obligó a utilizar un algoritmo de tipo *laberinto*. Como veremos posteriormente, este tipo de algoritmos realizan muchas exploraciones innecesarias, resintiéndose la velocidad de proceso. Por otra parte, la función de coste utilizada en este trabajo es

$$Coste = W_r xR + W_t xT \tag{2.7}$$

donde R es el número de redes que impiden completar el cableado del circuito, T es el retardo del camino crítico actualizado mediante un análisis, W_r *y* W_t son pesos para normalizar los diferentes términos de la función de coste. Además de los problemas de lentitud en la actualización del cableado comentados anteriormente, este algoritmo de ubicación y rutado simultáneos pierde efectividad con la función de coste dada por la Ecuación 2.7. En esta función de coste no existe una medida o estimación directa del cable necesario para rutar el circuito, por lo que se pierde capacidad de discriminación entre soluciones. Así, la función de coste no distingue entre dos ubicaciones a las que les falte el mismo número de redes por cablear (R), pero que tengan distintas demandas

de cable. Sin embargo, es muy probable que sea más beneficiosa aquella ubicación que hasta el momento haya demandado menos cable.

Algoritmos de ubicación

La ubicación es un problema NP-completo. Esta complejidad hace que sea conveniente la utilización de algoritmos probabilísticos para aproximarnos a la solución óptima. Dada la gran cantidad de posibles soluciones exploradas por estos algoritmos, el tiempo de cómputo necesario para determinar la calidad de las ubicaciones se convierte en un factor clave. Precisamente, una de las grandes ventajas de la métrica del *semiperímetro* como función de coste es su sencillez y rapidez de cálculo. Durante las últimas décadas, y debido al continuo incremento de la densidad de integración de los circuitos, se ha intensificado la investigación en los algoritmos de ubicación. Existe una serie de algoritmos que simulan fenómenos de la naturaleza como son: *Dirigido por Fuerzas, Enfriamiento Simulado, Simulación de la evolución o Algoritmos genéticos.* Otros, están basados en técnicas de partición como el algoritmo de *Corte-mínimo*, o técnicas de agrupamiento como el *Crecimiento de Clusters.* Finalmente, podemos destacar algoritmos analíticos como la *programación lineal* o la *programación cuadrática.* A continuación vemos una breve descripción de los más destacados.

El algoritmo *Dirigido por Fuerzas* [Qui75] simula un sistema de cuerpos enlazados por muelles. Es decir, aquellos bloques que están conectados por redes se supone que ejercen fuerzas atractivas entre ellos. La magnitud de estas fuerzas es directamente proporcional a la distancia entre los bloques. Si los bloques se dejarán libres, éstos se moverían hasta alcanzar una situación de equilibrio. Es decir, la solución para la ubicación de los bloques es aquélla en que el sistema alcanza el equilibrio. Por lo tanto, para calcular la posición de los bloques se plantean las ecuaciones de las fuerzas que intervienen en el sistema, y se resuelven utilizando los mismos métodos que se aplican a la mecánica clásica.

El algoritmo de *Simulación de la Evolución* [CP86] es similar al proceso natural de adaptación de las especies al entorno. El algoritmo comienza con un conjunto de configuraciones iniciales para la ubicación. Este conjunto de configuraciones iniciales

se generan de forma aleatoria y se les denomina población. Los individuos de la población consisten en cadenas de símbolos que representan posibles soluciones para el problema de la ubicación. El valor de los individuos se establece de acuerdo con una función de coste que estima su calidad. La población evoluciona mediante un proceso iterativo de generación en generación. En cada iteración se evalúa el *coste* de los individuos, y de acuerdo con su calidad, se asignan a los individuos probabilidades para ser seleccionados como padres. Mediante la aplicación de los operadores cruce y mutación sobre los padres seleccionados, se generan los nuevos individuos denominados descendencia. Los individuos que componen la descendencia se evalúan, y pasan a competir por la supervivencia con el resto de individuos de forma que el tamaño de la población permanece constante. La selección de los individuos basada en el coste, hace que los individuos de la población *mejoren* de generación en generación. Una de las claves para el éxito de los algoritmos evolutivos en la ubicación, es el diseño de un operador de cruce que sea rápido, y que tras su aplicación, no exija rehacer todo el cálculo de la función de coste. Esta dificultad, junto con la existencia de varios parámetros que necesitan ser ajustados para el buen funcionamiento del algoritmo, hace que existan pocas implementaciones de los algoritmos evolutivos aplicados a la resolución del problema de la ubicación.

En los algoritmos de ubicación basados en el *Corte-Mínimo* [Bre77], se realizan sucesivamente particiones del circuito en dos subcircuitos, minimizando el número de redes que pertenecen a las dos particiones simultáneamente. Por otra parte, en cada nivel de particionamiento, el área disponible del circuito integrado se divide alternativamente mediante líneas horizontales y verticales en dos subsecciones. En cada etapa de particionamiento, se asocian los subcircuitos con subsecciones. El proceso se repite hasta que cada subcircuito se compone de un sólo bloque lógico y tiene una posición definida. Entre las herramientas orientadas a FPGAs que utilizan esta técnica, podemos destacar la herramienta Altor [RSV85].

En las técnicas de *Crecimiento de Clusters* se parte de la ubicación de un bloque *semilla*. En cada iteración, se selecciona un bloque que esté fuertemente relacionado con los ya ubicados, y se sitúa cerca de ellos. La forma de selección, tanto del bloque

siguiente como de la posición donde éste se ubicará, hacen que el algoritmo genere soluciones de mayor o menor calidad, en más o menos tiempo.

En la *programación cuadrática* [Hall70], la ubicación de los bloques se determina a partir de la información obtenida de la *lista de redes*, y de las posible ubicaciones de los bloques. Así, en la asignación cuadrática la solución para el problema de la ubicación se encuentra hallando los *autovalores* y *autovectores* de una matriz construida a partir de la matriz de conectividad entre bloques, y de la matriz de distancia entre posibles ubicaciones de esos bloques. En circuitos de gran tamaño, y en el estilo de diseño de celdas standard, se han aplicado con éxito técnicas combinadas de particionamiento recursivo y programación cuadrática [KSJA91]. Los resultados producidos son comparables en calidad y tiempo a los generados por la técnica del *Enfriamiento Simulado* [SS95].

El algoritmo de *Enfriamiento Simulado* (*ES*) es uno de los métodos de ubicación más desarrollados hasta el momento. La efectividad del *ES* en la ubicación ha sido probada en diferentes trabajos [KGV83, SS84], y es utilizado por las herramientas de diseño físico comerciales [Tri93]. El *ES* es un algoritmo de mejora iterativa. Partiendo de una ubicación generada de forma aleatoria se realizan movimientos de intercambio en la posición de los bloques lógicos. Los movimientos son aceptados o rechazados dependiendo de la variación de coste producida. El movimiento se acepta siempre que se produce un decremento en el coste. En caso de que se produzca un incremento de coste, el movimiento se acepta con una probabilidad dada por el *factor de Boltzman* ($e^{-\Delta C/T}$), donde T es un parámetro denominado *temperatura*. Se suele escoger un valor inicial alto de este parámetro de forma que al principio casi todos los movimientos son aceptados. La *temperatura* se va disminuyendo según avanza la optimización reduciéndose, de esta forma, la probabilidad de aceptación de movimientos que produzcan incremento de coste. El ritmo de enfriamiento adecuado dependerá, en general, de varios factores como son la naturaleza y tamaño del problema considerado, así como la función de coste elegida. Entre las implementaciones del *ES* orientadas a la ubicación sobre FPGAs, podemos destacar la herramienta VPR (Versatile Place&Route) [BR97] por la calidad de los resultados que proporciona.

A pesar del éxito del *ES* en la ubicación, existen algunas características del algoritmo que deben tenerse en consideración cuando se afrontan variaciones del problema o se exploran nuevas funciones de coste. Los programas de enfriamiento son específicos, y generalmente no son aplicables directamente a otros problemas y funciones de coste. Debido a esta estrecha relación entre el programa de enfriamiento y el problema o la función de coste considerada, la investigación sobre nuevas funciones de coste para afrontar nuevos desafíos se convierte en una tarea compleja. Por una parte, se requieren costosos estudios experimentales para determinar un programa de enfriamiento adecuado. Por otra parte, en el estudio de nuevas funciones de coste, hay una carencia de soluciones de referencia para validar la adecuación de ese programa de enfriamiento a la función de coste tratada. Puede ser beneficioso, por lo tanto, la investigación de las relaciones subyacentes que pueden existir entre los distintos parámetros que indican el curso o desarrollo de la optimización, y el programa de enfriamiento que hacen que la optimización llegue a buen término. El desarrollo de un programa de enfriamiento auto-adaptable a distintos problemas y funciones de coste sería el objetivo último.

2.6.2 Cableado sobre FPGAs

Durante la etapa de cableado correspondiente al ciclo de diseño físico, se establecen las conexiones entre los bloques lógicos mediante la configuración de los recursos de cableado de la FPGA. Si bien este problema es similar al planteado en otros dispositivos como los arrays de puertas y celdas standard, aquí presenta características particulares; por una parte, la anchura de los segmentos de canal es muy limitada, por lo tanto, si no se resuelven los problemas de congestión, el circuito no podrá implementarse en la FPGA [EMHB95]. Por otra parte, los segmentos de canal están unidos entre sí a través de interruptores programables, los cuales incrementan los retardos de las conexiones. Debido a su complejidad combinatoria, el problema del cableado usualmente se resuelve en dos etapas: cableado global y cableado detallado. En la fase de cableado global se asignan segmentos de canal de la FPGA para la conexión de cada red (Figura 2.4) de forma que se cumplan las restricciones en cuanto a la capacidad de los canales. Como veremos más adelante, en el caso de redes

multiterminal como la de la Figura 2.4, el problema es equivalente a la determinación del *"Árbol de Steiner Mínimo Rectilíneo"*, con la particularidad de que la capacidad de los segmentos de canal no debe ser sobrepasada durante el cableado de las redes.

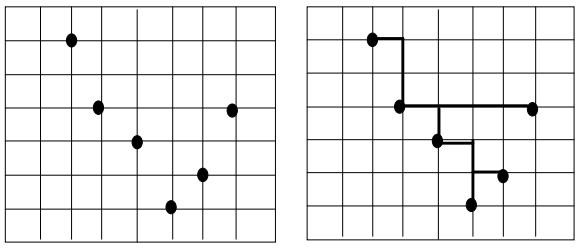


Figura 2.4 *Cableado global de una red multiterminal (la retícula sobre la que se realiza el cableado se corresponde con los segmentos de canal de la FPGA)*

Podemos distinguir dos tipos de algoritmos para el cableado global atendiendo a cómo se realiza el tratamiento de las redes multiterminal. Por una parte tenemos los algoritmos que descomponen las redes multiterminal en redes de dos terminales. Por otra parte, están los algoritmos que afrontan directamente el tratamiento de las redes multiterminal. Entre los primeros podemos destacar los algoritmos de *"El camino más corto"*, algoritmos de tipo *"laberinto"*, y los algoritmos de *"línea de prueba"*. Entre los segundos, destacan algunos algoritmos de tipo *"laberinto"* orientados a redes multiterminal, y distintas aproximaciones al "*Árbol de Steiner Mínimo Rectilíneo "*.

Los algoritmos de *"El camino más corto"* seleccionan el conjunto de vértices que dan lugar a un camino de coste mínimo para una conexión de dos terminales. En los algoritmos de *"línea de prueba"* [MT68, Hig69] se persigue reducir los requisitos de memoria. Así, el camino recorrido por una red a través de varios segmentos de canal se describe mediante una línea que los englobe, en lugar de detallar todos los segmentos de canal que la componen. En los algoritmos de tipo *"laberinto"* [Lee61, Sou78, Had75], en una primera fase de exploración, se inician y expanden simultáneamente varios caminos desde el nodo fuente hasta que alguno alcanza el nodo destino. En una segunda fase, se identifica el camino que ha tenido éxito. Existen diversas versiones de este algoritmo, tanto para redes de dos terminales como para redes multiterminal. Las versiones para redes multiterminal producen soluciones de calidad, de forma que es uno

de los métodos más utilizados en la actualidad. Sin embargo, el cableado se hace demasiado lento debido a la gran cantidad exploraciones innecesarias.

El problema que surge en el cableado global de una red multiterminal, es equivalente al problema del *"Árbol de Steiner Mínimo"*. Éste es un problema NP-completo, y se puede definir como sigue: dado un grafo G(V,E) y un subconjunto $D \subseteq V$, seleccionar un subconjunto $V' \subseteq V$, de forma que $D \subseteq V'$, y tal que V' induce un árbol de coste mínimo. A los elementos del conjunto D se les denomina *puntos de demanda*, mientras que los elementos del conjunto V'-D constituyen los puntos de Steiner [She99]. *Al Árbol de Steiner Mínimo* para una métrica *rectilínea* se le denomina *Árbol de Steiner Mínimo Rectilíneo (ASMR)*. Este problema es también un problema NP-completo. Existe una interesante relación entre el *ASMR* y el *Árbol de Expansión Mínima* (*AEM*) [Hwa76] que viene dada por la Ecuación 2.8.

$$\frac{L_{AEM}}{L_{ASMR}} \leq \frac{3}{2} \tag{2.8}$$

Como consecuencia de esta relación, muchas aproximaciones para resolver el *ASMR* están basadas en el cálculo del *AEM* [LBH76, HVW90, KZ97, MVG99]. De la misma forma, el *AEM* se utiliza habitualmente como referencia para comparar las distintas aproximaciones al *ASMR*. Así, el porcentaje de mejora M sobre el *AEM* se puede determinar mediante la Ecuación 2.9.

$$M = \frac{L_{AEM} - L_{ASMR}}{L_{AEM}} x100 \tag{2.9}$$

Hasta la fecha, ha destacado la aproximación *Batched Iterated 1-Steiner* (BI1S) [GRSZ94, Rob] por la calidad de las soluciones obtenidas, encontrándose éstas, en valores medios, a un 0.5% de los valores óptimos [MVG99]. Sin embargo, los tiempos de cómputo de éste y otros algoritmos como el IRV [MVG99], basado en programación lineal entera, son todavía excesivos para redes con un número elevado de nodos. La

lentitud de estos algoritmos se ha puesto de manifiesto recientemente con el desarrollo del algoritmo *Geosteiner* [WWZ98] que permite el cálculo del *Árbol de Steiner Mínimo Rectilíneo* de forma exacta en tiempos comparables a los de BI1S.

La segunda etapa que permite completar el rutado del circuito es la del cableado detallado (Figura 2.5). Durante esta fase se especifica qué cables, pertenecientes a cada segmento de canal, se utilizan para la conexión de cada red con las restricciones impuestas por el cableado global. El problema del cableado detallado normalmente se resuelve de forma incremental. El orden de rutado de las regiones se establece siguiendo diferentes factores como la pertenencia de una determinada red al camino crítico, o el número de redes que pasan a través de una región.

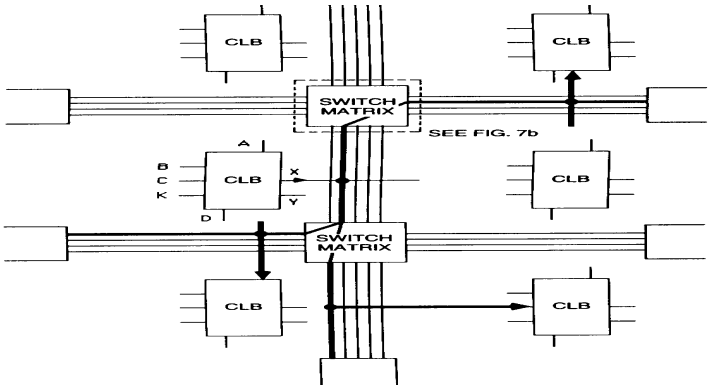


Figura 2.5 *Cableado detallado de una red. Se especifica qué cables de cada segmento de canal se utilizan en la conexión. (*R.Xilinx)

Por último, debemos mencionar la técnica de cableado de *"ripup and reroute"* [CLWL95, BR97], que es complementaria de cualquiera de las anteriores. Esta técnica se aplica habitualmente cuando no es posible rutar más redes debido al agotamiento de los recursos de interconexión, bien durante el cableado global o bien durante el cableado

detallado. Su objetivo es aliviar la congestión levantando las redes que entran en conflicto, y variando el orden en el que son rutadas.

Entre los trabajos publicados para el cableado de circuitos sobre FPGAs podemos citar una serie de herramientas como son CGE, SEGA, TRACER-fpga, GBP, OGC, IKMB, FPR, y VPR.

CGE (Coarse Ghaph Expansion) [Bro92, BRV92] fue una de las primeras herramientas de cableado detallado orientada a FPGAs. El algoritmo está compuesto de dos fases. En una primera fase determina las distintas alternativas de cableado y su coste. En una segunda fase, elige una alternativa para cada conexión de acuerdo con una función de coste. SEGA [LB93] es una mejora de la herramienta CGE, ya que considera aspectos como la velocidad del circuito rutado además de su rutabilidad. TRACER-fpga [CLWL95, LW97] es una herramienta que resuelve los conflictos entre redes aplicando una estrategia de *rip-up and re-route* guiada por una técnica de optimización basada en *Simulación de la Evolución.* GBP [WM94] (Greedy Bin-Packing) implementa una estrategia de búsqueda local para realizar el cableado global y detallado. Su sucesor OCG [WM95, WM97] (Orthogonal-Coupled Greedy Heuristics) desarrolla una técnica para salir de valores óptimos locales basada en el acoplamiento de dos algoritmos de búsqueda local. En IKBM [AR96] se obtiene una aproximación al *Graph Steiner Tree* (GST) para resolver el problema del cableado detallado. FPR (FPGA Placement and Routing) [ACGR98] se basa en un particionamiento recursivo para realizar una ubicación y rutado global simultáneos. Por último, VPR [BR97] (Versatile Place&Route) realiza una mejora del algoritmo *"Pathfinder Negotiated Congestion"* para realizar el cableado del circuito.

Si bien, los detalles de cada algoritmo se pueden encontrar en las referencias reseñadas, es importante destacar la superioridad mostrada por el algoritmo VPR sobre todos los demás. Así, la Tabla 2.3 reproduce los resultados publicados en [BR97], donde se realiza una comparación de la anchura de canal demandada para cablear un conjunto de circuitos de prueba con estas herramientas. Con el fin de uniformizar los experimentos, la ubicación de los circuitos se llevó a cabo con el algoritmo ALTOR (excepto para FPR que realiza una ubicación y rutado simultáneos).

Tabla 2.3 *Comparación de la anchura de canal demanda por distintos algoritmos de rutado*

CIRC.	*CGE*	*SEGA*	*FPR*	*GBP*	*OGC*	*IKBM*	*TRACER*	*VPR*
9symml	9	9	9	9	9	8	6	6
alu2	12	10	10	11	9	9	9	8
apex7	13	13	9	11	10	10	8	8
example2	18	17	13	13	12	11	10	9
k2fix	19	16	17	17	16	15	14	12
term1	10	9	8	10	9	8	7	7
vda	14	14	13	13	11	12	11	10

En la Tabla 2.3 se observa la superioridad de VPR para el cableado de circuitos sobre FPGAs con respecto a las otras herramientas de cableado. La anchura de canal demandada por VPR es siempre igual o inferior a la anchura demandada por todas las demás herramientas. Estas diferencias son todavía más pronunciadas cuando la ubicación del circuito se realiza también con la herramienta VPR como veremos en el capítulo 6. Como ya se ha señalado, *VPR* realiza la ubicación con el algoritmo de *Enfriamiento Simulado*, guiado por la función de coste *Bounding-Box*. Para el cableado recurre a una modificación del algoritmo *Pathfinder negotiated congestion* [EMHB95], el cual está basado en técnicas de "*laberinto*" y "*ripping-up and re-routing*". Debido a la calidad de las soluciones proporcionadas por VPR, a lo largo de este trabajo de investigación se utilizará esta herramienta, como referencia para evaluar la calidad de los métodos y estrategias desarrollados.

2.7 Arquitectura de los Sistemas Multi-FPGA

Cuando un circuito sobrepasa la capacidad de una única FPGA, parece razonable recurrir a varias FPGAs para implementar del circuito. A un dispositivo compuesto por un conjunto de FPGAs y un sistema de interconexión entre ellas se le denomina sistema

Multi-FPGA. En los últimos años han ido apareciendo diferentes tarjetas Multi-FPGA [Guc2000] con el objetivo de proporcionar sistemas digitales reconfigurables de gran capacidad. Estos proyectos han cambiando de acuerdo con las distintas tecnologías de FPGAs disponibles en cada momento. Por ejemplo, el proyecto PAM (*Programmable Active Memories*) [Ber92][PAM2000] ha ido evolucionando a través de distintas tarjetas Multi-FPGA como *PAM Proto* (1987), *PeRLe-0* (88), *DECPeRLe-1* (92), *Tc-Pamette* (94), *PCI Pamette* (96), *FireLink* (97). También podemos encontrar otros proyectos dirigidos a diversas aplicaciones como *Splash* y *Splash2 (Standford Parallel Applications for Shared-Memory)* [Gok90][SWG91] orientadas a la supercomputación, *Space* y *Space2* (*Scalable Parallel Architecture for Concurrency Experiments*) ideadas para explorar diseños de proceso en paralelo, *MORRPH (MOdular and Reprogrammble Real-time Processing Hardware)* [DKTC95] dirigida al procesamiento de imagen, o *Transmogrifier* [Gal94] *y Transmogrifier2* [LGIRC98] cuyo objetivo es proporcionar entornos para el prototipado rápido. Entre todos estos sistemas Multi-FPGAs, los hay implementados sobre uno [Gal94] o varios circuitos impresos [TBDHA94, Quic98]. También se han propuesto sistemas Multi-FPGAs construidos sobre *Módulos Multi-Chip (MCMs)* [Dob92, DGI94, Dar95, Lan95]. En este caso, las FPGAs y la red de interconexión se montan sobre un substrato reduciéndose el área y la potencia consumida, mientras mejora la velocidad. Sin embargo, para que estos dispositivos sean comercialmente viables, se requiere dar solución a algunos aspectos como la densidad de interconexiones y la disipación térmica.

A pesar del gran número de sistemas Multi-FPGA existente, se puede decir que el principal inconveniente que presentan en la actualidad todos los Sistemas Multi-FPGA es la escasez de *pines de entrada/salida* de las FPGAs, lo cual restringe fuertemente la capacidad de comunicación entre ellas. Esta carencia, obliga a recurrir a grandes sistemas Multi-FPGA para soportar las demandas de comunicación, a costa de provocar bajos porcentajes de utilización de CLBs. Por lo tanto, la topología de interconexión entre las FPGAs, que afecta a la capacidad de comunicación, se presenta como un elemento clave para el éxito de estos sistemas. A pesar de las diferencias entre los distintos sistemas, es posible establecer una clasificación de éstos en dos grandes grupos

atendiendo a la topología de la red de interconexión dispuesta entre sus componentes. Así, podemos distinguir entre *topología de barras cruzadas* y *topología de malla*. En las topologías de barras cruzadas, las FPGAs suelen estar interconectadas a través de componentes de interconexión programables como los FPICs (*Field Programmable Interconect Components*) o FPIDs (*Field Programmable Interconect Devices*) (Figura 2.6). Estos dispositivos se pueden programar para proporcionar diferentes configuraciones de interconexión entre FPGAs. Cuando la conectividad entre FPGAs es total, el sistema de interconexión de barras cruzadas se denomina *completo*, mientras que en caso contrario se denomina *parcial*. El sistema de *barras cruzadas parcial* suele ser el tipo de conexionado más común por razones de coste. Podemos encontrar diversos sistemas Multi-FPGAs que utilizan este tipo de interconexión como el sistema Transmogrifier-2, donde las FPGAs están conectadas entre sí por un sistema de barras cruzadas parcial. En otras configuraciones como CHAMP [Guc2000] se utilizan FPGAs para implementar el sistema de barras cruzadas.

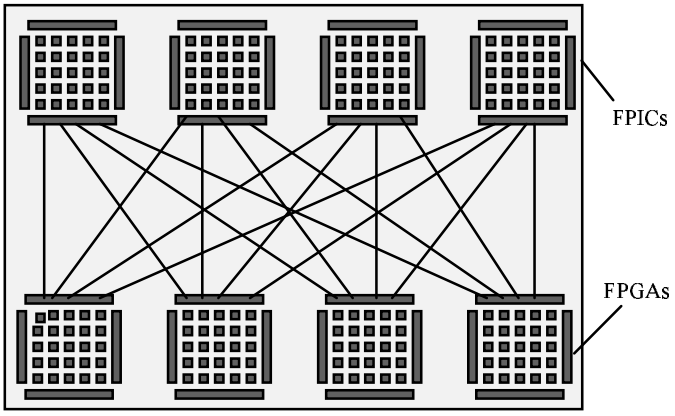


Figura 2.6 *Topología de Barras Cruzadas*

La principal ventaja de un sistema de interconexión de barras cruzadas es que permite equilibrar la capacidad de comunicación entre las distintas FPGAs. Otra ventaja es que los FPICs pueden disponerse de forma jerárquica para proporcionar mayor conectividad en sistemas grandes [Quic98], aún a costa de mayores tiempos de retardo. Entre los inconvenientes de los sistemas de barras cruzadas está el alto coste de los

dispositivos FPICs. Además, el uso de estos dispositivos lleva asociado un incremento de área y tiempo de retardo en la comunicación entre FPGAs.

La otra forma básica de conectar las FPGAs en un sistema Multi-FPGA es la topología de malla. En este caso, las FPGAs están directamente conectadas a sus vecinas más próximas (Figura 2.7). A lo largo de los años han aparecido diferentes sistemas con topología malla. Así, en el proyecto PAM se desarrollaron diferentes mallas fijas como el sistema *PeRLe-0* de 25 *FPGAs*, o el sistema *TBc-Pamette* formado por una matriz de 2x2 FPGAs. Otros sistemas construidos con topología de malla se describen en [SM93, TBDHA94].

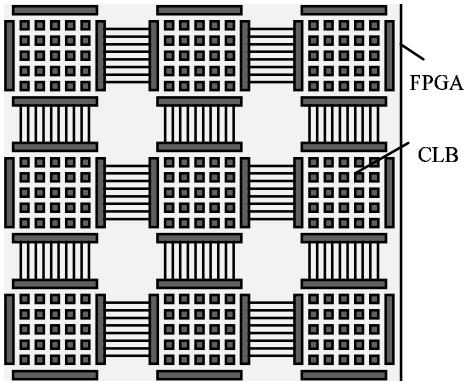


Figura 2.7 *Topología malla*

La topología de malla despierta un gran interés debido al ahorro de área que supone la sencillez de la red de interconexión entre las FPGAs. Otra ventaja adicional es la facilidad de expansión simplemente incrementado el tamaño de la red. La arquitectura de malla proporciona una buena conectividad local por lo que se ha utilizado con éxito en diversas aplicaciones como la criptografía, el análisis de imagen o la física de altas energías. Sin embargo, la topología de malla sufre problemas de conectividad y de tiempos de retardo en aplicaciones que exigen una fuerte comunicación entre FPGAs no adyacentes. Con el objeto de mejorar la capacidad de conexión entre FPGAs se han propuesto mallas alternativas [Hau94]. Así, en la malla de 8-caminos las FPGAs no sólo se conectan a sus vecinas Norte, Sur, Este y Oeste, sino también a sus vecinas en dirección diagonal. Otra alternativa propuesta es una malla donde parte de las

conexiones dirigidas anteriormente a FPGAs vecinas, se dedican para la conexión con FPGAs más lejanas. Si bien, se han realizado algunos estudios teóricos apoyando estás alternativas [HBE98], faltan evidencias experimentales con circuitos reales sobre su idoneidad. Dentro de la categoría de mallas, y como una simplificación de estas, podemos clasificar también a los arrays lineales de FPGAs. Entre estos sistemas, podemos citar al sistema Splash cuya efectividad se ha demostrado en aplicaciones de tipo sistólico [Gok91]. Sin embargo, los arrays de FPGAs de una dimensión tienen limitaciones para otros tipo de aplicaciones con mayores requisitos de conectividad. Finalmente, las mallas tridimensionales despiertan especial interés por la elevada capacidad lógica que se espera obtener de ellas [SDP92].

Existe también un elevado número de sistemas híbridos [Kha99], donde se trata de aprovechar las bondades de las dos topologías básicas de interconexión descritas hasta el momento. De la misma forma, con el objetivo de salvar las fuertes carencias de conexionado entre FPGAs, muchos sistemas optan por el uso de buses. Así, el sistema *DECPeRLe1* combina una malla de 4x4 con buses globales compartidos.

2.8 Diseño físico sobre Sistemas Multi-FPGA

Como hemos visto en la sección anterior, los sistemas Multi-FPGA han surgido con el objetivo de proporcionar grandes sistemas digitales reconfigurables. Sin embargo, esa gran capacidad lógica no tiene ningún beneficio si no se le puede sacar rendimiento de forma adecuada. La escasez de recursos de interconexión de los sistemas Multi-FPGA se está convirtiendo en su gran cuello de botella, conduciendo, en muchos casos, a porcentajes de utilización lógica menores del 20% [HBE98]. La complejidad y el tamaño creciente de los circuitos implementados sobre sistemas Multi-FPGA, exige la automatización de los procesos implicados en el diseño físico con el objetivo de adecuar la demanda de interconexión a los recursos disponibles. A continuación vemos las distintas etapas que componen el ciclo de diseño físico orientado a sistemas Multi-FPGA. Puesto que el objetivo de estudio de esta tesis son los sistemas Multi-FPGA con topología de malla, en la descripción de cada etapa del diseño físico, se presta especial

atención en resaltar las diferentes problemáticas que presentan las topologías de malla y de barras cruzadas. Estas dos topologías se pueden considerar los polos opuestos entre los que se sitúan las topologías híbridas que presentarán una amalgama de problemas asociados a ambas topologías.

El ciclo de diseño físico sobre sistemas Multi-FPGA se divide habitualmente en varias etapas: *Partición, Ubicación global, Cableado global, Asignación de pines, y Ubicación y Rutado local de las FPGAs*. En la partición, la lista de CLBs lógicos correspondiente al circuito se divide en bloques, de forma que la funcionalidad de cada bloque se pueda implementar sobre una FPGA.

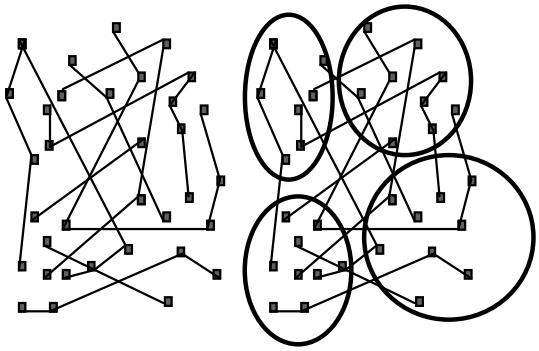


Figura 2.8 *Partición de CLBs lógicos en FPGAs lógicas*

En los últimos años se han realizado diferentes estudios para la partición de sistemas sobre arquitecturas Multi-FPGA. En estos estudios se aprecian diversas orientaciones que responden a distintos niveles, métodos y objetivos de la partición de sistemas sobre arquitecturas Multi-FPGA. Así, podemos encontrar estudios de particionamiento a nivel de descripción de comportamiento [LGSV2000], a nivel de lista de CLBs [OG95], o a nivel de puertas lógicas [Hau94]. También se han desarrollado distintos métodos que cubren un amplio abanico de soluciones como son el particionamiento jerárquico [OHG96], el particionamiento recursivo [KB95], el particionamiento espectral [CSZ94], o el particionamiento basado en algoritmos genéticos [Hid99]. El particionamiento temporal orientado a sistemas reconfigurables también ha sido objeto de numerosos estudios [PV99, LW99].

Como se comentó en la sección anterior, el reducido número de *pines de entrada/salida* que presentan las FPGAs limita fuertemente el aprovechamiento íntegro de la capacidad lógica de los sistemas Multi-FPGA. El principal objetivo de las herramientas de partición que trabajen con estas limitaciones debe ser, por lo tanto, minimizar la comunicación entre particiones para adecuarla a la capacidad física real de conexionado entre FPGAs. En los sistemas de *barras cruzadas* puros, la capacidad de comunicación entre todas las FPGAs está equilibrada. Es decir, existen canales de igual tamaño para la comunicación de cada FPGA con todas las demás. De esta forma, la capacidad de comunicación entre FPGAs alejadas físicamente, es la misma que la capacidad de comunicación entre FPGAs vecinas. Por lo tanto, para estos sistemas se puede realizar una partición del circuito en N subcircuitos (siendo N el número de FPGAs disponibles), equilibrando la demanda de comunicación entre particiones. El objetivo de esta partición consistirá en no sobrepasar la capacidad de conexión entre FPGAs que ofrece esta topología. La mayoría de los métodos de particionamiento desarrollados hasta la fecha consideran que la capacidad de comunicación entre dos FPGAs cualesquiera es la misma, y por tanto son aplicables a esta topología. Sin embargo, cuando se aplican estos métodos a topologías Multi-FPGA de tipo malla se obtienen resultados poco convincentes. Ello explica que, hasta donde conocemos, no existan estudios publicados sobre partición de circuitos orientada a mallas de FPGAs puras, donde la partición realizada se haya puesto a prueba con las fases de ubicación y cableado global, y donde además se considere un conjunto amplio de circuitos de prueba.

Podemos aventurar que el futuro de la topología de malla pasa (mientras la capacidad de comunicación entre FPGAs no se incremente) por una cuidadosa asignación de sus escasos recursos de comunicación. Es en este aspecto, donde las herramientas de partición deben incorporar, de alguna forma, el *conocimiento* acerca de las características peculiares de esta topología. Más concretamente, deben ser capaces de hacer previsiones de recursos para la comunicación entre FPGAs distantes. El mismo razonamiento se puede aplicar, si bien en menor grado, para las topologías híbridas. El éxito de una topología determinada va a depender, no sólo de sus virtudes intrínsecas,

sino de que se incluya ya en la etapa de partición, el *conocimiento* acerca de los recursos reales de cableado. Este conocimiento permitirá a las herramientas realizar una asignación cuidadosa de las redes del circuito sobre los recursos de cableado disponibles en cada arquitectura. Por lo tanto, para las topologías híbridas será también conveniente realizar una transformación de herramientas genéricas de partición, a herramientas dirigidas a topologías concretas.

Una vez realizada la partición del circuito, en la fase *ubicación global* se lleva a cabo la correspondencia entre los bloques lógicos o particiones, y las FPGAs físicas del sistema (Figura 2.9). Para realizar la correspondencia entre particiones y FPGAs físicas, se debe establecer alguna métrica de *proximidad* que recoja las características específicas de cada arquitectura. Por ejemplo, en arquitecturas híbridas esta métrica debe reflejar la mayor o menor dificultad de comunicación entre dos FPGAs cualesquiera. Una vez definida esta métrica, se puede recurrir a las mismas técnicas de optimización que las utilizadas para resolver el problema de ubicación sobre FPGAs, teniendo en cuenta que en este caso los nodos que componen las redes son FPGAs en lugar de CLBs .

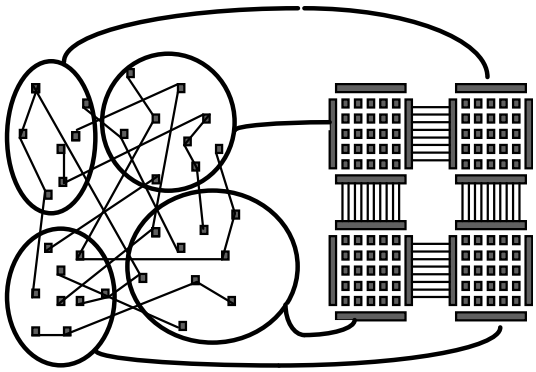


Figura 2.9 *Ubicación global*

Un aspecto importante a tener en cuenta, es que la etapa de ubicación global queda difuminada en el caso de las dos topologías extremas consideradas. En el caso de un sistema de barras cruzadas puro, esta fase no es necesaria pues cualquier asignación

entre particiones y FPGAs físicas es equivalente. En el caso de mallas de FPGAs puras, donde no todas las FPGAs tienen la misma capacidad de comunicación (piénsese por ejemplo en las FPGAs externas de la malla respecto de las internas), ya en la etapa de partición se debería haber considerado esta circunstancia, y por lo tanto la posición a ocupar por cada partición ya debería estar bastante delimitada por la etapa de particionamiento. Es más, con el objetivo de hacer frente a las fuertes carencias de comunicación entre FPGAs distantes sería conveniente, en esta topología, integrar las etapas de partición y ubicación en un único proceso.

Una vez asignadas las particiones a sus correspondientes FPGAs físicas, durante las etapas de *rutado global* y de *asignación de pines* se asignan recursos concretos de interconexión entre FPGAs, para la comunicación entre particiones que estén relacionadas. El rutado global debe repartir, de forma equilibrada, las redes del circuito entre los distintos recursos de interconexión.

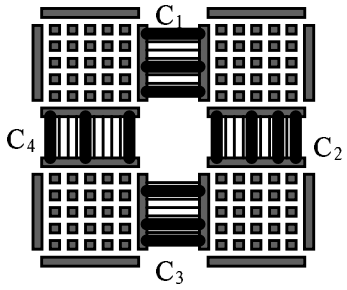


Figura 2.10 *Cableado global y Asignación de pines*

En el caso de los sistemas Multi-FPGA con topología de malla como el de la Figura 2.7, no existen conexiones directas entre todas las FPGAs. Así, la comunicación entre FPGAs no vecinas se debe realizar a través de otras FPGAs. De nuevo, en sistemas Multi-FPGA con topología de malla, y a diferencia de la topología de barras cruzadas, el éxito de la fase de cableado estará fuertemente condicionada por la previsión de recursos que hayan realizado las herramientas de partición y ubicación para la comunicación particiones asociadas a FPGAs distantes. Esta previsión de recursos se

debe realizar en la etapa de particionamiento, que es donde se decide qué particiones se tendrán que comunicar.

Finalmente, en las últimas etapas del ciclo de diseño físico, se realiza la *ubicación* y el *rutado local* de los bloques lógicos sobre las FPGAs individuales cumpliendo las restricciones impuestas por la *asignación de pines*. Es decir, preservando las posiciones de *entrada/salida* asignadas a los extremos de las redes que enlazan las particiones. Para realizar esta tarea se puede recurrir a herramientas de ubicación y rutado dirigidas a FPGAs individuales.

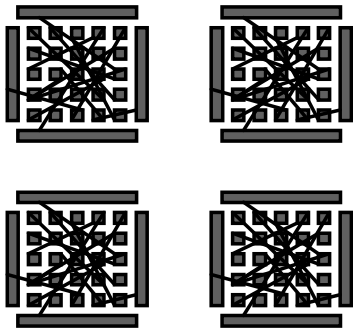


Figura 2.11 *Ubicación y rutado de las FPGAs individualmente*

Como hemos visto a lo largo de esta sección, el ciclo de diseño físico sobre sistemas Multi-FPGA puede variar enormemente con el tipo de topología a la cual esta dirigido. Por lo tanto, los resultados obtenidos mediante una metodología asociados a un determinado sistema Multi-FPGA, en general no son extrapolables para otras topologías.

Capítulo 3

Ubicación en FPGAs mediante *Optimización Natural*

3.1 Ubicación mediante *Enfriamiento Simulado*

La ubicación es una de las fases que más tiempo consumen del ciclo diseño físico de circuitos integrados, afectando su calidad tanto al área como a la velocidad de los circuitos. En el caso de las FPGAs, debido a que los recursos de cableado son limitados, la ubicación puede incluso comprometer el éxito del rutado, y por lo tanto, la implementación del circuito en una FPGA determinada. La ubicación es un problema NP-completo, de forma que continuamente surgen nuevos desafíos debido al creciente número de elementos lógicos que integran las FPGAs comerciales. Como se comentó en el capítulo anterior, para resolver el problema de la ubicación sobre FPGAs debemos trabajar en dos direcciones. Por una parte, tenemos que cuantificar los objetivos mediante una función de coste que permita la discriminación entre las distintas soluciones. Por otra parte, y dado que el número de soluciones crece considerablemente con el tamaño de la FPGA, es necesaria la utilización de un algoritmo de optimización combinatoria que realice una exploración adecuada del espacio de soluciones. En este capítulo nos centraremos en este segundo aspecto.

Entre los algoritmos de optimización combinatoria, el *Enfriamiento Simulado (ES)* [KGV83] destaca por ser uno de los algoritmos que mayor éxito ha tenido en la resolución del problema de la ubicación hasta el momento [She99]. El *ES* se puede considerar un método de optimización combinatoria general. Se ha aplicado con éxito a una gran variedad de problemas derivados de la ciencia y de la ingeniería. Así, se encuentran aplicaciones del algoritmo en diversas áreas como el análisis de imágenes, las redes neuronales, la visualización gráfica, el tratamiento de datos o el diseño VLSI [WLL88]. De igual forma que otros algoritmos de búsqueda local o mejora iterativa, el *ES* lleva a cabo una búsqueda dirigida en el espacio de soluciones. Así, mediante una

sucesión de perturbaciones realizadas a partir de una solución inicial, se aceptan aquellas perturbaciones que nos acercan a la solución de coste mínimo. A diferencia de otros métodos de búsqueda local, la naturaleza probabilística del *ES* le permite aceptar temporalmente soluciones peores que la actual para escapar de valores mínimos locales.

El *ES* se basa en la analogía que se puede encontrar entre el proceso de recocido realizado en sólidos para alcanzar su estado de energía mínima, y la resolución de problemas de optimización combinatoria. El proceso de recocido de sólidos consiste en su calentamiento a altas temperaturas, y su posterior enfriamiento gradual de forma que se alcance el estado de mínima energía. El estado de mínima energía del sólido sólo se alcanza si la temperatura máxima es suficientemente alta, y si el enfriamiento se realiza de forma suficientemente lenta. Para la resolución de problemas de optimización combinatoria se considera un proceso similar. En este caso, en lugar de estados físicos del sistema se habla de soluciones del problema de optimización combinatoria, y en lugar de energía de un estado se habla del coste de una solución [AK89]. A partir de estas equivalencias, el proceso de recocido se puede simular mediante tres operadores o funciones:

Operador de generación -. Es un mecanismo para generar una nueva solución por medio de transformaciones locales sobre la solución actual. Este operador permite moverse por el espacio de soluciones para explorar las distintas alternativas. El operador generación se debe diseñar de forma que el espacio de soluciones sea muestreado eficientemente. En algunos casos, a partir de un espacio de soluciones dado, se define el concepto de vecindad entre soluciones. En estos casos, se puede decir que el operador generación permite la transformación de una solución en otra solución vecina. En este mismo sentido, en algunos problemas se puede definir el concepto de distancia entre soluciones de acuerdo con el tipo de transformación realizada para pasar de una solución a otra. Recurriendo a este concepto, se puede comenzar la búsqueda aplicando transformaciones grandes (de larga *distancia*), e ir reduciendo paulatinamente el grado de las transformaciones según avanza la búsqueda. Mediante esta técnica, en algunas ocasiones, se puede mejorar la velocidad de optimización del algoritmo.

Probabilidad de aceptación -. Una vez aplicado el operador generación de nuevas soluciones, estás se someten a un test de aceptación. La probabilidad de aceptación se establece de acuerdo con el factor de Boltzman (Ecuación 3.1). Es decir, los movimientos que generan soluciones con coste menor son siempre aceptados, mientras que los movimientos que implican un incremento en el coste se aceptan con una probabilidad que varía de forma exponencial decreciente con magnitud del incremento, y que depende de un parámetro T, denominado *temperatura* por su equivalencia con el caso del recocido en sólidos. Esta probabilidad de aceptación va a permitir al algoritmo escapar de valores mínimos locales.

$$P_{acept} = \begin{cases} 1 & \Delta C \leq 0 \\ e^{-\frac{\Delta C}{T}} & \Delta C > 0 \end{cases} \qquad (3.1)$$

donde T es la temperatura y ΔC es la variación de coste debida a la transformación local

Programa de enfriamiento.- Con objeto de que la optimización converja hacia un valor mínimo global, el parámetro temperatura que controla la probabilidad de aceptación se va disminuyendo según avanza el proceso de búsqueda. Por lo tanto, el programa de enfriamiento define el método con el que se selecciona la temperatura inicial y el ritmo de enfriamiento. El programa de enfriamiento es un elemento clave para el éxito de la optimización. Así, si se selecciona un valor muy bajo para la temperatura inicial, la búsqueda puede quedar estancada en un valor mínimo local, comprometiendo el éxito de la optimización. Por el contrario, si la temperatura inicial es muy alta, se puede estar realizando una búsqueda completamente aleatoria durante cierto tiempo reduciéndose la eficiencia de la optimización. El ritmo de enfriamiento es también clave para el resultado de la optimización. Un enfriamiento rápido puede estancar la búsqueda en valores óptimos locales, mientras que un enfriamiento demasiado lento provoca exploraciones innecesarias o redundantes.

De los tres operadores del *Enfriamiento Simulado*, sólo la probabilidad de aceptación se puede considerar independiente del problema. Como veremos posteriormente, en algunos estudios teóricos se han propuesto distintas densidades de probabilidad para los parámetros que controlan el operador generación de soluciones, asociadas a determinados programas de enfriamiento para garantizar la convergencia del algoritmo hacia el valor óptimo global. Sin embargo, debido a que la configuración de las soluciones puede ser muy diversa, el operador generación tiene que ser diseñado de forma específica para cada problema. En cuanto al programa de enfriamiento existen una infinidad de propuestas. El resto de esta sección, lo dedicamos a analizar algunas de estas alternativas, y a vislumbrar las características que serían deseables para el programa de enfriamiento.

Uno los fundamentos teóricos del *Enfriamiento Simulado*, es que el enfriamiento se debe realizar de forma cuasiestática. Es decir, el sistema debe estar muy cerca del *equilibrio térmico* durante todo el proceso de optimización. En la práctica, para aproximarse a esta propiedad en el proceso enfriamiento, normalmente se suele actuar en dos frentes. Por una parte, se realizan pequeños decrementos en la temperatura de forma que la ruptura del equilibrio no sea grande. Por otra parte, se lleva a cabo un número grande de movimientos en cada temperatura de manera que se restablezca el equilibrio. Por lo tanto, se puede entrever la existencia de un compromiso en el proceso de optimización entre estos dos parámetros: magnitud de los decrementos de temperatura, y número de movimientos realizados en cada temperatura. Así, decrementos pequeños en la temperatura necesitarán menos movimientos para restablecer el equilibrio, mientras que decrementos grandes en este parámetro requerirán un mayor número de movimientos para restablecerlo. El ajuste de estos dos parámetros puede ser complejo y requerir auxilio de una mano experta.

Aunque existen estudios teóricos basados en las cadenas de Markov [LM86, Haj88, AK89] que demuestran la convergencia asintótica del algoritmo de *ES* hacia el valor óptimo global, ese carácter asintótico de la demostración, predice infinitas transformaciones para garantizar el objetivo. Por ejemplo, el programa de enfriamiento $T=T_o/\ln(t)$, combinado con el uso de la distribución Gaussiana-Makoviana en el

operador generación de soluciones, garantiza la convergencia hacia el valor óptimo global [GG84]. Sin embargo, este programa de enfriamiento resulta demasiado lento, y además sólo garantiza la convergencia cuando el número de transformaciones t, tiende a infinito. En la práctica, no siempre es posible permitirse el lujo de alcanzar la solución óptima. Los problemas reales de optimización tienen una restricción de tiempo para su resolución. Por lo tanto, el objetivo se debe centrar en encontrar una buena solución dentro los márgenes de tiempo permitidos. Así, existen otros programas de enfriamiento más rápidos como los programas de enfriamiento exponenciales ($T_{t+1} = cT_t$, $0 < c < 1$), ($T_t = T_0 \exp(at)$), cuyo uso está muy extendido, o el denominado *Fast Annealing* (*FA*) [SH87] que utiliza la distribución de Cauchy para el operador generación, y la fórmula $T=T_o/t$ para la disminución de la temperatura.

Aunque el *ES* se ha aplicado con éxito a mucho problemas de optimización combinatoria, son necesarios una gran cantidad de estudios experimentales para determinar el mejor programa de enfriamiento en cada caso. Pequeñas variaciones del problema o incluso de la función de coste normalmente requieren la adaptación de los parámetros del algoritmo. Así, la temperatura inicial y el programa de enfriamiento, que afectan al compromiso entre la velocidad y la calidad de las soluciones generadas, se deben ajustar experimentalmente para funciones de coste nuevas. Se han realizado algunos intentos de automatizar la adaptación del *ES* a nuevos problemas y funciones de coste [BS86][Ing89][Ing96]. Así, el algoritmo *Adaptive Simulated Annealing (ASA)* [Ing96] trata de sintonizar automáticamente el algoritmo para diversos problemas incluyendo funciones de coste multidimensionales. A diferencia de los métodos anteriores, la distribución utilizada por ASA para la generación de nuevas soluciones, tiene en cuenta tanto el carácter finito de las dimensiones de los problemas reales, como las distintas sensibilidades de los parámetros asociados a cada dimensión. Su programa de enfriamiento asociado a la dimensión i es de la forma

$$T_i(t)=T_{0i} \exp(-c_i t^{1/D})$$

donde T_{0i} y c_i son parámetros ajustables a cada problema y D la dimensión del espacio con el que se trabaja. Sin embargo, a pesar de proporcionar una fórmula simple para el

programa de enfriamiento, de nuevo aparece un conjunto de parámetros que necesitan ser ajustados para cada problema como son T_{0i}, c_i y D.

En todos los programas de enfriamiento vistos hasta el momento, la temperatura depende del número de transformaciones evaluadas, es decir del tiempo. Sin embargo, no existe ninguna variable que relacione la temperatura con el problema o la función de coste, excepto una serie de parámetros que se deben ajustar experimentalmente para cada problema. De acuerdo con la relación dada por la Ecuación 3.1, la probabilidad de aceptar una nueva solución con un incremento en el coste depende de ΔC y T. Puesto que en estos programas de enfriamiento, T no está relacionado de forma directa con la función de coste, existe una carencia de información para evaluar el beneficio o perjuicio de aceptar una nueva solución con incremento en el coste. Por lo tanto, cada función de coste requiere un programa de enfriamiento exclusivo, o un ajuste experimental de distintos parámetros.

Existe otro tipo de programas de enfriamiento donde la temperatura es función de la solución (x) y de su coste, en lugar de ser función del número de soluciones exploradas t. Éste es el caso del *Enfriamiento Simulado Generalizado* [BS86], donde la temperatura depende del coste f de la solución explorada x, según la expresión $T(x)=(f(x))^g$, donde g es una constante a ajustar. En el caso de que el mínimo global se encuentre en cero, la temperatura será pequeña cerca del mínimo global, y el algoritmo convergerá hacia ese valor. Para el caso más común en el que el mínimo no es cero, es necesario algún tipo de transformación para que el algoritmo converja. La transformación más directa, sería una simple traslación $T(x)=(f(x)-f(x_m))^g$, siendo $f(x_m)$ el valor del mínimo. Sin embargo, el valor del mínimo normalmente es desconocido, lo que limita las posibilidades de aplicación del método.

Hasta el momento, hemos hecho un repaso de algunos de los programas de enfriamiento más conocidos. Todas estas variaciones del *ES* producen excelentes resultados cuando se particularizan para un problema determinado mediante estudios experimentales. Sin embargo, estas aproximaciones no presentan la flexibilidad suficiente para proporcionar determinadas características de adaptabilidad en sus parámetros y estrategias como son:

1) Programa de enfriamiento auto-sintonizado .- Los estudios realizados para determinar el programa de enfriamiento para un problema habitualmente no son aplicables a otros problemas. Es más, los cambios en la función objetivo afectan de forma importante al programa de enfriamiento que se debe aplicar. Por lo tanto, sería deseable el desarrollo de un programa de enfriamiento fácil de ajustar a diversos problemas, o lo que es mejor, que se ajuste a sí mismo.

2) Temperatura de una solución.- Dado un problema particular, la temperatura inicial para el *ES* se ve fuertemente afectada por la función de coste. De forma habitual, esta temperatura se escoge arbitrariamente con el único requisito de que sea suficientemente alta para evitar convergencias prematuras de algoritmo. Sería deseable manejar el concepto de *"temperatura de una solución"* con el objetivo de que la temperatura inicial se pudiera determinar automáticamente a partir del coste de la solución de partida. La optimización en varias etapas se podría beneficiar de este concepto. Por ejemplo, se podría realizar una primera fase de optimización con la técnica más rápida disponible, y cuando esta técnica no pudiera proporcionar nuevas mejoras, una segunda fase con el algoritmo de *ES* podría tomar el relevo. De la misma forma, varias funciones de coste se podrían ordenar de acuerdo con el compromiso *tiempo/calidad* que proporcionan para guiar al algoritmo de ES en diversas etapas, y de esta forma acelerar el proceso de búsqueda. Así, una función de coste poco precisa pero fácil de calcular, y por lo tanto de cómputo rápido, puede descartar soluciones inferiores. Optimizaciones posteriores guiadas por funciones de coste más precisas, podrían refinar la búsqueda. Ambas aproximaciones demandan el cálculo de la temperatura de la solución proporcionada por la primera etapa. Una temperatura inicial sensiblemente más alta de su valor idóneo, puede echar a perder las mejoras logradas en la primera etapa, mientras que una temperatura más baja podría hacer que el *ES* perdiese oportunidades de mejora.

El concepto de temperatura de una solución facilitaría también el diseño incremental. Para pequeñas modificaciones realizadas en el diseño del circuito, la ubicación previa, que estaría cercana al óptimo, podría aprovecharse puesto que la temperatura inicial se adaptaría a esa solución de partida.

3) Flexibilidad en el compromiso tiempo/calidad.- El tiempo de computo disponible y la calidad demandada para las soluciones, dependen de manera importante del problema particular de optimización que se está tratando. La flexibilidad en el compromiso tiempo/calidad puede ser el planteamiento adecuado para acomodar el algoritmo de optimización a la potencia de cómputo actual, y adaptarlo fácilmente cuando esa potencia mejore. Así, sería deseable disponer de un único parámetro para adaptar el programa de enfriamiento al tiempo disponible, y a la calidad demandada para las soluciones.

En este capítulo se presenta el algoritmo de *"Optimización Natural"*, un método nuevo auto-sintonizable para el tratamiento de problemas de optimización combinatoria, que persigue las características de adaptabilidad descritas anteriormente.

3.2 Optimización Natural

En esta sección se presenta el algoritmo de *Optimización Natural* (*ON*), un método adaptativo para la optimización de la ubicación. Una de las tareas que más tiempo consumen en el ajuste de los parámetros del algoritmo de *Enfriamiento Simulado*, es la determinación de un regla para calcular la temperatura inicial y su programa de enfriamiento. El objetivo principal de la *ON* es automatizar el ajuste de esos parámetros.

En el *ES*, una solución se acepta con una probabilidad que viene dada por el factor de Boltzman. De esta forma, se establece una equivalencia entre costes y energías. Sin embargo, no se proporciona ninguna equivalencia para la temperatura y se supone que la disminución de este parámetro debe ser determinada experimentalmente para cada problema. En *ON*, nosotros sugerimos la relación $T \sim |\overline{\Delta C}|$ para el cálculo de la temperatura. Así, la probabilidad de aceptar una nueva solución en el método de *Optimización Natural* será

$$P_{acept} = \begin{cases} 1 & \Delta C \leq 0 \\ e^{-\frac{\Delta C}{k|\overline{\Delta C}|}} & \Delta C > 0 \end{cases} \tag{3.2}$$

donde $\overline{\Delta C} = \frac{C_{mínimo\ actual} - C_{inicial}}{movimientos}$, es el decremento de coste medio alcanzado hasta el momento, mientras k es un parámetro que permite obtener diferentes compromisos de *tiempo/calidad*. Es importante destacar que $C_{mínimo\_actual}$ no es el coste de la solución actual, sino el menor coste logrado hasta el momento. De acuerdo con esta definición, el parámetro T se auto-sintoniza durante todo el proceso de enfriamiento, de forma que ahora no nos tenemos que preocupar acerca del programa de enfriamiento. En el método de *Optimización Natural*, la probabilidad de aceptar una nueva solución con un incremento de coste depende de la historia de la optimización mantenida en $\overline{\Delta C}$. De esta forma, la probabilidad de aceptación obtiene información acerca de las características de la función de coste y del espacio de soluciones durante la optimización.

Grado de retroceso

El parámetro k de la ecuación 3.2 se puede denominar *grado de retroceso* y controla la probabilidad de aceptar un movimiento con un incremento de coste igual al decremento de coste promedio por iteración logrado hasta el momento actual. Como $\overline{\Delta C}$ es el decremento de coste promedio por iteración, $k \cdot \overline{\Delta C}$ es el decremento de coste promedio obtenido en k iteraciones. Entonces, cuando se acepta un movimiento con un incremento de coste igual a $k \cdot |\overline{\Delta C}|$, podemos decir que el proceso de búsqueda de soluciones está retrocediendo k iteraciones.

El valor de k se puede ajustar a la calidad demandada y al tiempo de cómputo disponible. Además, su valor se puede acomodar fácilmente a futuras mejoras en la capacidad de cálculo para obtener soluciones de mayor calidad. La Figura 3.1 muestra un ejemplo de optimización con *ON*. El problema consiste en la optimización del circuito de prueba e64.net [BR96b] sobre una FPGA guiado con la función de coste *Bounding Box* (BB) (Capítulo 2), la cual proporciona una estimación normalizada de la longitud de cable total. Las gráficas muestran la relación entre el coste y el número de soluciones evaluadas para diferentes valores del parámetro k. Se puede observar en la figura que valores del parámetro k mayores proporcionan mejores soluciones pero más lentas.

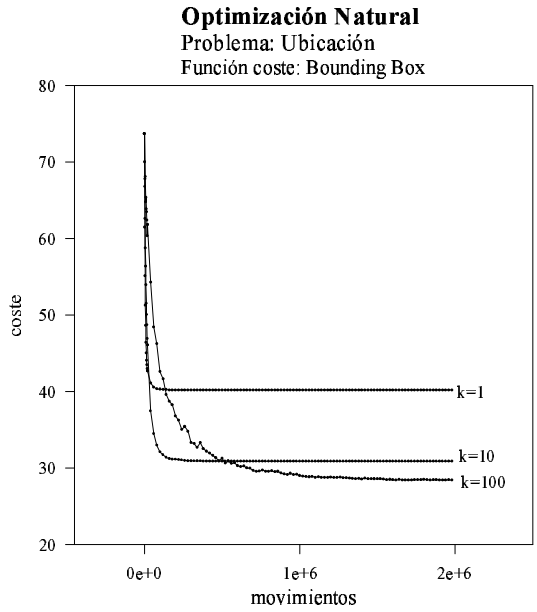


Figura 3.1 *Comparación de la optimización de la ubicación con el método de Optimización Natural para diferentes valores de k.*

En general, el tiempo de optimización depende de diversos factores como son: la eficacia en la generación de nuevas soluciones, la complejidad en el cómputo de la función de coste o la capacidad de cálculo disponible. Experimentalmente, hemos observado que el tiempo de optimización también es directamente proporcional al valor de k. Por lo tanto, para adecuar el valor de k al tiempo disponible t para la optimización, podemos determinar experimentalmente el tiempo de optimización t_1 asociado al valor k=1 (el cual dependerá de los factores antes señalados), y aplicar la relación $k=t/t_1$.

Temperatura de una solución

En el *Enfriamiento Simulado* la temperatura inicial es un parámetro importante que se debe ajustar de forma precisa [SBD87]. Si la temperatura inicial está por debajo de su valor exacto, la obtención del óptimo global se puede ver comprometida. Por otra parte, si la temperatura inicial es demasiado alta, se puede estar realizando una búsqueda sin sentido durante varias iteraciones. De la misma forma, si la búsqueda comienza a partir de una buena solución inicial que proviene de una optimización parcial realizada con

otro método, un valor de la temperatura inicial por encima de su valor exacto podría echar a perder la *bondad* de la solución de partida [RKW90].

El concepto de *"temperatura de una solución"* permitiría salvar estas preocupaciones. En *ON*, y de acuerdo con la Ecuación 3.2, la temperatura de una solución se define como el promedio del decremento de coste logrado cuando se alcanza esa solución. Este decremento de coste promedio depende del valor de k, por lo tanto se pueden definir distintas temperaturas de una solución según el valor de k. Pero dado un valor de k, la temperatura de una solución debe tener el mismo valor sin tener en cuenta el camino descrito o las soluciones intermedias exploradas para alcanzarla.

Si tenemos en cuenta que el decremento de coste promedio no tiene un valor significativo hasta que se acepta por los menos un movimiento, la temperatura no se puede determinar para la solución inicial o de partida, pero si se podrá determinar para las siguientes soluciones aceptadas. Para salvar esta dificultad, la temperatura se puede fijar inicialmente en un valor muy bajo de forma que sólo los movimientos que producen un decremento en el coste sean aceptados. Una vez que un primer movimiento con decremento de coste es aceptado, $\overline{\Delta C}$ se hace distinto de cero. A partir de ese momento, la temperatura será sintonizada automáticamente a su valor correcto durante todo el proceso de "*optimización natural*".

La Figura 3.2 muestra un ejemplo de lo expuesto anteriormente. En este experimento, se optimiza la ubicación del circuito e64.net con el algoritmo de *ON* y la función de coste *BB* para un valor de k=100. El eje X representa la temperatura en escala logarítmica, mientras que el eje Y representa el coste de las soluciones intermedias exploradas en el proceso de optimización. Se puede observar que partiendo inicialmente de tres soluciones aleatorias caracterizadas por una temperatura muy baja (*puntos a, b, c*), y después de unas pocas iteraciones de adaptación, el proceso de optimización evoluciona en los tres casos por la misma curva *coste/temperatura*. Por lo tanto, se puede concebir el concepto de *"temperatura de una solución"*. Dado un valor de k, la temperatura de una solución es única independientemente del camino descrito para alcanzarla. Para apoyar esta idea, la misma figura también muestra la optimización de la ubicación partiendo de dos soluciones a medio camino parcialmente optimizadas,

y caracterizadas igualmente por una temperatura inicial muy baja (*puntos d, e*). En ambos casos, la temperatura es igualmente auto-sintonizada al valor que le corresponde en unas pocas iteraciones. Además, los procesos de optimización recorren el mismo camino (coste=f(T)) que en los ejemplos anteriores.

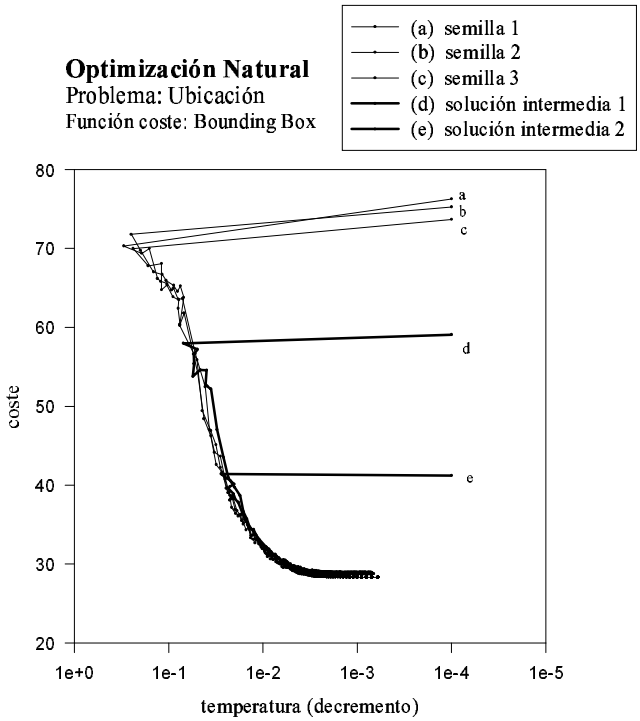


Figura 3.2 *Relación Coste/Temperatura para la optimización de la ubicación con el algoritmo de Optimización Natural. A partir de diferentes soluciones iniciales, todas las optimizaciones evolucionan por la misma curva de Coste/Temperatura.*

Generación de nuevas soluciones

En la generación de nuevas soluciones para someterlas al test de aceptación, y de cara a mejorar la eficacia en la exploración del espacio de soluciones, puede ser valioso tener en cuenta el momento en que se encuentra el proceso de optimización. Como se comentó en la sección 1, algunas aplicaciones del *ES* estrechan la *magnitud* de las transformaciones conforme avanza la búsqueda de soluciones. En el problema de la ubicación se puede aplicar esta técnica. Llamaremos rango de los movimientos (*R*) a la máxima distancia permitida entre dos bloques de la FPGA que se intercambian en un

momento dado para generar una nueva solución. Dado que ΔC depende de *R*, la probabilidad de aceptación se verá afectada por *R* y *T*. Por lo tanto, podría ser razonable sintonizar *R* con *T* para alcanzar altos porcentajes de aceptación de soluciones. Por ejemplo, una relación lineal entre *R* y *T* estrecharía el rango de los movimientos conforme la búsqueda avanza. La Ecuación 3.3 muestra esta relación

$$R = R_{min} + \alpha \cdot (T - T_{final}) \tag{3.3}$$

donde se ha exigido que al final de la optimización ($T=T_{final}$), el rango de los movimientos coincida con la mínima distancia a la que se pueda mover un bloque (R_{min}). Estableciendo la condición de que al principio de la optimización ($T=T_{inicial}$) no haya restricciones en los movimientos ($R=R_{max}$) tendremos

$$R_{max} = R_{min} + \alpha \cdot (T_{inicial} - T_{final}) \tag{3.4}$$

despejando α

$$\alpha = \frac{R_{max} - R_{min}}{T_{inicial} - T_{final}} \tag{3.5}$$

Por lo tanto, α se puede calcular en tiempo de ejecución después de algunos movimientos de adaptación que permitan "determinar" el valor de la temperatura inicial asociado a la solución de partida (Figura 3.5).

3.3 Enfriamiento Simulado vs Optimización Natural

En esta sección se comparan los resultados producidos por los métodos de *Enfriamiento Simulado* y *Optimización Natural* en la ubicación para un conjunto de circuitos de prueba. El *ES* es uno de los métodos que mayor éxito ha obtenido en la optimización de la ubicación hasta la fecha. *Versatile Place&Route* (VPR) [BR97] es un ejemplo de una herramienta cuidadosamente sintonizada para realizar la ubicación y el rutado de circuitos en FPGAs utilizando el *ES*. La optimización de la ubicación se lleva a cabo por medio de la minimización de la función de coste *BB*. La bajada de temperaturas ha sido cuidadosamente estudiada de forma que los parámetros del enfriamiento se ajustan automáticamente a diferentes tamaños de circuito. Se debe tener en cuenta que

esta aplicación del *ES* incorpora las ideas formadas durante varios años a raíz de diferentes trabajos [HRS86][SS90].

Por el contrario, la bajada de temperaturas de la *ON* es muy simple. La probabilidad de aceptar una nueva solución viene dada por la Ecuación 3.2. El programa de enfriamiento se auto-sintoniza de la forma descrita en la sección 2 de este capítulo. El valor de k se ha fijado en 100 para asegurar la calidad de las soluciones obtenidas. La Figura 3.3 muestra el proceso de optimización para el circuito *e64.net* llevada a cabo por ambos métodos: *ES* y *ON*. El eje X se representa el número de movimientos evaluados, mientras que el eje Y representa el coste de cada solución. Debemos destacar que, puesto que el tiempo invertido por los dos métodos en la evaluación de cada movimiento es el mismo, el número de iteraciones o movimientos evaluados es proporcional al tiempo de optimización. Como se puede observar en la gráfica, el coste con la *ON* decae mucho más rápido de como lo hace en el *ES*, mientras se obtiene la misma solución final.

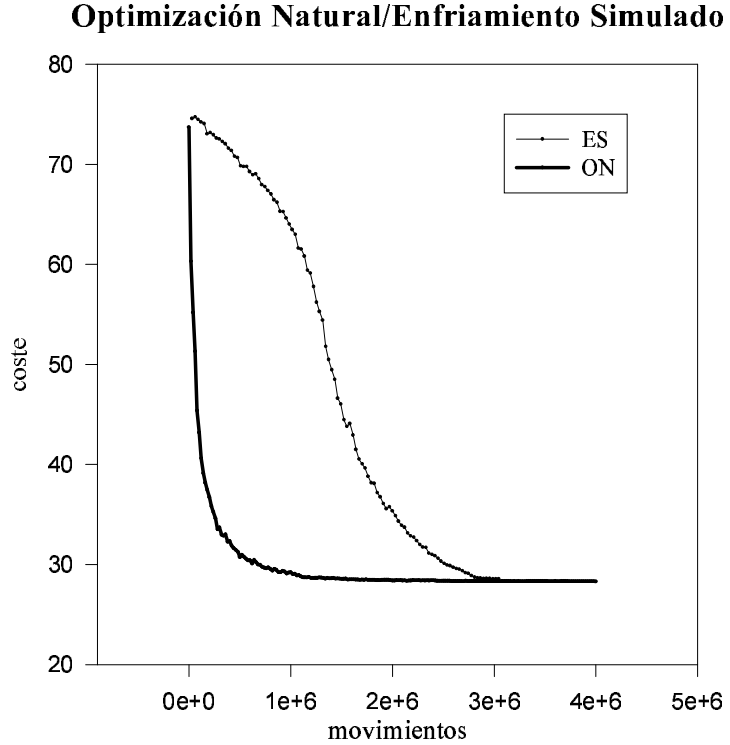


Figura 3.3 *Comparación de la relación coste/movimientos para la optimización con ES y ON en el problema de la ubicación sobre FPGAs*

Otra gráfica interesante para el estudio de ambos algoritmos es la relación entre el coste y la temperatura de las soluciones exploradas. La Figura 3.4 muestra las gráficas *coste/temperatura* para los procesos de optimización con *ES y ON*. El eje X representa la temperatura para el *ES* y kT para la *ON,* ambos en escala logarítmica, mientras que el eje Y representa el coste de cada solución. Sorprendentemente, después de seguir caminos iniciales distintos, las dos curvas se juntan en una curva única. Este resultado apoya el concepto de *"temperatura de una solución"* introducido anteriormente. Los mismos puntos de las gráficas *coste=f(T)* son encontrados por medio de dos métodos opuestos: uno artificial (*ES*) y el otro más natural (*ON*).

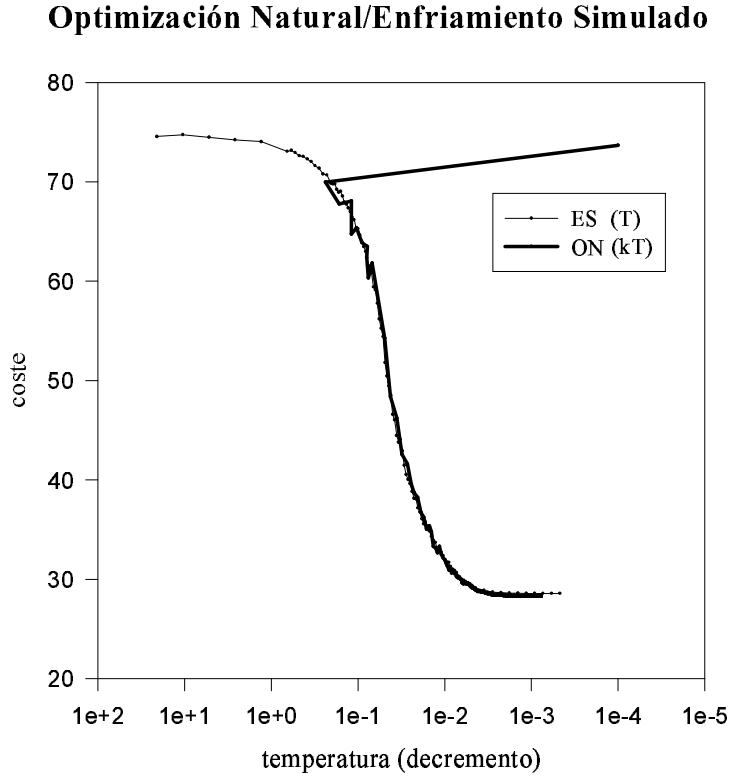


Figura 3.4 *Evolución de la relación coste/temperatura para las ubicaciones con ES y ON*

Debido a que la *ON* adapta continuamente la temperatura al desarrollo de la optimización, con esta técnica no se realizan movimientos sin obtener *provecho* como

los que se llevan a cabo en el *ES*, y por lo tanto, la temperatura decrece mucho más rápidamente que con el *ES* (Figura 3.5).

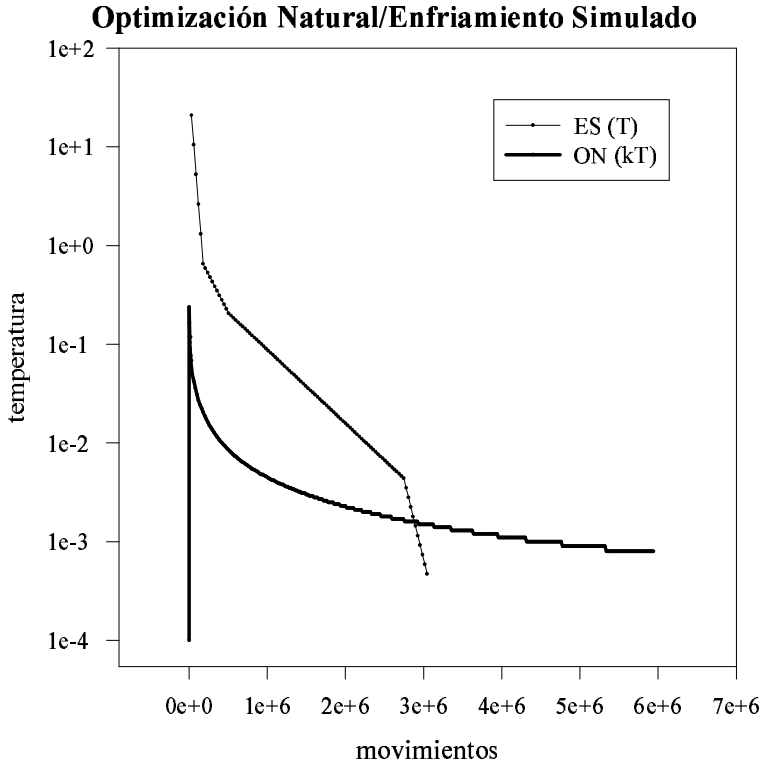


Figura 3.5 *Evolución de la relación temperatura/movimientos para las ubicaciones con ES y ON*

La comparación entre los dos algoritmos se ha extendido para un conjunto de circuitos de prueba [BR97]. La Tabla 3.1 muestra las características de los circuitos y el tamaño de las FPGAs necesarias para albergarlos. El valor de k para la *Optimización Natural* se ha fijado en 160 para asegurar una calidad para las soluciones comparable a la proporcionada por *VPR*.

Tabla 3.1 *Circuitos de prueba: CLBs e IOBs*

CIRC	#CLBs	#I/O	Tamaño FPGA
alu4	1522	22	40x40
apex2	1878	41	44x44
apex4	1262	28	36x26
bigkey	1707	426	54x54
diffeq	1497	103	39x29
dsip	1370	426	54x54
ex5p	1064	71	33x33
frisc	3556	136	60x60
misex3	1397	28	38x28
s298	1931	10	44x44
seq	1750	76	42x42
tseng	1047	174	33x33

La Tabla 3.2 muestran los valores de la función de coste *BB* optimizada por *ES* y *ON*, así como el número de movimientos realizado. Las últimas dos columnas muestran una comparación relativa en calidad (*Q*) y tiempo (*T*) de los resultados producidos por *ON* respecto a los producidos por *ES*, medida como sigue:

$$Q = 100 \cdot \frac{BB(SA) - BB(ON)}{BB(inicial) - Min(BB(SA), BB(ON))} \qquad \textit{(Calidad)}$$

$$T = 100 \cdot \frac{Movimientos(ON) - Movimientos(SA)}{Max(Movimientos(SA), Movimientos(ON))} \qquad \textit{(Tiempo)}$$

El factor *Q* cuantifica la relevancia de una diferencia de coste en las soluciones proporcionadas por los métodos *ES* y *ON*, respecto al total optimizado. De la misma forma *T* mide diferencias significativas en los tiempos de optimización respecto a la

magnitud de estos tiempos. Se debe tener en cuenta que valores positivos de *Q* y valores negativos de *T* implican una mejora del *ON* respecto del *ES* y viceversa. Puesto que el valor de |Q| es menor del *±1*% para todos los circuitos, podemos decir que la calidad lograda por ambos métodos es similar, mientras que en los tiempos de cómputo la *ON* mejora o iguala al *ES* en diez de los doce casos.

Tabla 3.2 *Comparación ON/ES para un conjunto de circuitos de prueba*

CIR.	Coste (Bounding Box)			Movimientos (x10^6)		Q	T
	Inicial	*ES*	*ON*	*ES*	*ON*	%	%
alu4	612	192	192	21.2	15.0	+0.0	-29
apex2	912	267	266	28.6	26.8	+0.2	-06
apex4	503	180	180	16.3	16.3	+0.0	-00
bigkey	1085	187	187	35.7	34.7	+0.0	- 03
diffeq	670	146	145	22.5	16.5	+0.2	-26
dsip	895	171	171	28.5	36.1	+0.0	+21
ex5p	427	162	162	13.8	9.4	+0.0	-32
frisc	2288	516	516	72.4	75.9	+0.0	+05
misex3	588	188	188	18.8	18.0	+0.0	-04
s298	750	203	203	29.3	26.9	+0.0	-08
seq	804	247	246	26.7	23.0	+0.2	-14
tseng	420	93	93	15.5	9.4	+0.0	-39

3.4 Conclusión

Uno de los principales inconvenientes del *ES* para resolver problemas de ubicación, consiste en la gran cantidad de estudios experimentales necesarios para adaptar los parámetros del algoritmo a nuevas funciones de coste. En general este inconveniente proviene de la falta de relación entre el parámetro de la temperatura que controla la

probabilidad de aceptar una nueva solución, y el momento en que se encuentra la optimización.

En este capítulo hemos presentado el algoritmo de *Optimización Natural*, que proporciona un método nuevo para el tratamiento de problemas de ubicación. Una simple modificación del método ampliamente utilizado del *ES*, lo convierte en un método más adaptativo. En la *ON*, la probabilidad de aceptar una solución se calcula incorporando información acerca las características de la función de coste, obtenida mientras se realiza la optimización. Así, la temperatura se auto-sintoniza continuamente con el desarrollo de la optimización.

Estas características de adaptabilidad permiten a la *ON* obtener los resultados de calidad del *ES*, proporcionando además, la versatilidad necesaria para facilitar el estudio de nuevas funciones de coste y estrategias en el tratamiento de problemas de ubicación.

Capítulo 4

Ubicación en FPGAs mediante

Optimización Combinatoria Termodinámica

4.1. Introducción

En el Capítulo 3, se presentó el método de *Optimización Natural* (*ON*) para la aproximación a la solución de problemas de ubicación. En este capítulo se desarrolla el método de *Optimización Combinatoria Termodinámica* (*OCT*). El método *OCT* consiste en una fórmula para la actualización de temperaturas en el *Enfriamiento Simulado* (*ES*), derivada de principios termodinámicos. De la misma forma que la *ON*, el programa de enfriamiento se autosintoniza para distintos problemas y funciones de coste mientras mejora su rendimiento.

Como se vio en el Capítulo 3, uno de los fundamentos teóricos del *ES*, consiste en alcanzar el equilibrio en cada temperatura durante el proceso de enfriamiento. Para lograr este objetivo, normalmente se actúa en dos frentes. Por un lado, se controla el valor de la temperatura de forma que sus descensos sean pequeños. Por otro lado, se lleva a cabo un cierto número de movimientos a cada temperatura para recuperar el equilibrio. El ajuste de esos dos parámetros normalmente se realiza experimentalmente.

El método *OCT* proporciona una alternativa distinta para realizar el enfriamiento cerca del equilibrio, sin requerir costosos ajustes experimentales. El método *OCT* se deriva de los principios de la Termodinámica a través de la definición de procesos reversibles. La característica principal de los procesos reversibles es precisamente que los estados intermedios de la transformación son también estados de equilibrio. Además, puesto que la eficiencia de una transformación reversible mejora cualquier otra transformación realizada irreversiblemente, un enfriamiento del sistema realizado reversiblemente, teóricamente consigue la máxima eficiencia. En optimización

combinatoria, se puede interpretar esta propiedad como que un proceso reversible logra el compromiso calidad/tiempo óptimo, o que alcanza el máximo rendimiento.

En la siguiente sección se introducen distintas definiciones relacionadas con la Termodinámica necesarias para la presentación del método. En la sección 4.3 se describe el método *OCT*. En la sección 4.4 se realizan comparaciones con los resultados presentados en el Capítulo 3, y en la sección 4.5 se extraen las conclusiones.

4.2. Termodinámica

La Termodinámica clásica se centra en el estudio de las transformaciones correspondientes a fenómenos físicos macroscópicos relacionados con el calor y la temperatura [Agu84]. Su campo de aplicación se restringe a estados de equilibrio y transformaciones que se pueden representar mediante una serie continua de estados de equilibrio. Un sistema se dice que está en equilibrio termodinámico cuando se alcanza un estado donde las variables macroscópicas que caracterizan el sistema, alcanzan un valor constante independiente del tiempo.

La Termodinámica se fundamenta en tres postulados o principios, a partir de los cuales se derivan todas las demás leyes termodinámicas. En el primer principio se establece la relación entre el calor y el trabajo a través de la conservación de la energía. El segundo principio determina la dirección de los procesos termodinámicos y define el estado de equilibrio de los sistemas físicos. El tercer principio fija un límite para los valores de la temperatura y la entropía. En Termodinámica también se habla de un principio cero que define el concepto de equilibrio termodinámico.

La Termodinámica estudia los fenómenos físicos desde el punto de vista macroscópico basándose en las propiedades observables. Así, no se hacen hipótesis acerca de la estructura del sistema, ni estudia las interrelaciones microscópicas entre los componentes internos del sistema. Un sistema se define como una parte del espacio y su contenido, y la descripción del estado del sistema se realiza por medio de los valores de unas propiedades macroscópicas denominadas variables o coordenadas termodinámicas. Puesto que las variables termodinámicas tienen un significado macroscópico, sólo tiene sentido definirlas para sistemas formados por un gran número de componentes. Al

conjunto de variables independientes necesario para caracterizar el sistema se denomina variables de estado. El resto de las variables del sistema se representa como función de ellas. Una función que puede expresarse por medio de variables de estado se le denomina función de estado del sistema (p.e. energía interna, entropía, etc.). La variación de una función de estado depende solamente de los estados inicial y final, y no de los procesos que conectan los dos estados. Las magnitudes que no cumplen esta condición no son variables de estado (p.e. calor y trabajo).

Procesos termodinámicos

Un sistema experimenta un proceso o una transformación termodinámica cuando alguna de las variables que caracteriza el sistema cambia en el tiempo. Los estados inicial y final se consideran estados de equilibrio. Un proceso es cuasiestático cuando los estados intermedios son también estados de equilibrio. El proceso es reversible cuando una ligera variación en las condiciones externas hace cambiar el sentido de una transformación cuasiestática. Y el proceso es irreversible cuando los estados intermedios no son estados de equilibrio. Por lo general, un proceso reversible tiene las siguientes características:

1) Una pequeña variación sobre las condiciones del sistema cambia el sentido del proceso.
2) Es infinitamente lento.
3) Una transformación reversible es más eficiente que cualquier otra transformación realizada de forma irreversible.
4) En general no es realizable.

Principios termodinámicos

1) El primer principio de la Termodinámica establece la relación entre la variación de la energía interna (ΔU), el calor intercambiado (Q), y el trabajo realizado (W) en una transformación mediante la Ecuación 4.1.

$$\Delta U = Q - W \tag{4.1}$$

Si los estados inicial y final son estados infinitamente próximos, la Ecuación 4.1 se convierte en

$$dU = \delta Q - \delta W \tag{4.2}$$

2) El segundo principio determina la dirección de los procesos termodinámicos y el estado de equilibrio de los sistemas físicos. La expresión matemática del segundo principio viene dada por

$$\oint \frac{\delta Q}{T} \leq 0 \tag{4.3}$$

siendo $\oint \frac{\delta Q}{T} = 0$ para procesos reversibles y $\oint \frac{\delta Q}{T} < 0$ para irreversibles.

Por lo tanto, existe una función de estado del sistema denominada Entropía (S), cuya variación en un proceso reversible desde un estado inicial 1 a un estado final 2 es igual a

$$S_2 - S_1 = \int_1^2 \frac{\delta Q}{T} \tag{4.4}$$

Una consecuencia directa de la Ecuación 4.4, es que la variación de Entropía en un ciclo reversible es siempre nula. Si los estados inicial y final son infinitamente cercanos, la Ecuación 4.4 se convierte en

$$dS = \frac{\delta Q_R}{T} \tag{4.5}$$

y cuando el proceso es irreversible las Ecuaciones 4 y 5 se expresan como

$$S_2 - S_1 > \int_1^2 \frac{\delta Q}{T} \tag{4.6}$$

y

$$dS > \frac{\delta Q_I}{T} \tag{4.7}$$

respectivamente.

Variación de Entropía del Universo

Si se considera el universo, no hay intercambio de calor (δQ=0). Entonces, de acuerdo con las Ecuaciones 4 y 6 tenemos

$$S_2 - S_1 \geq 0 \tag{4.8}$$

Es decir, la Entropía del universo tiende a alcanzar un valor máximo. Se puede particularizar esta expresión para procesos reversibles. Imagínese un proceso reversible donde el sistema absorbe un calor δQ_R del exterior. La variación de Entropía del sistema es $\frac{\delta Q_R}{T}$. Puesto que el proceso es reversible las temperatura interna y externas coinciden, y la variación de Entropía de los alrededores es $-\frac{\delta Q_R}{T}$. Por lo tanto, $dS_{total}=0$ para procesos reversibles. Se debe destacar que aunque $dS_{total}=0$ para el proceso completo, la variación de Entropía del sistema y de los alrededores no es cero.

Orden y Entropía

De forma habitual, la Entropía se considera como una medida del desorden de los sistemas. Puesto que la variación de Entropía total es $S_2 - S_1 \geq 0$, existen muchos procesos que producen un incremento de Entropía en los sistemas, y de esta forma los sistemas se desordenan. Sin embargo, existen otros procesos donde los sistemas se ordenan. Por ejemplo, los cristales surgen espontáneamente de una solución de

moléculas e iones aleatoriamente distribuida, los lagos se congelan, y las estructuras de los animales surgen de una mezcla confusa de moléculas complejas. La Entropía de estos sistemas decrece mientras se ordenan. Aunque todos estos procesos ocurren espontáneamente, no ocurren de forma aislada, sino relacionados con otros sistemas aumentando la Entropía total.

Interpretación estadística de la Entropía

El objetivo de la Mecánica Estadística es interpretar y predecir las propiedades macroscópicas de los sistemas, a partir de las propiedades de las partículas que los constituyen. En Mecánica Estadística, cada estado termodinámico definido por magnitudes macroscópicas como la energía interna, la presión y la temperatura, corresponde a un gran número de microestados diferentes. La probabilidad termodinámica de un sistema (W), se define como el número de microestados que corresponde a un estado macroscópico. Existe una relación entre la Entropía S de un sistema y la probabilidad termodinámica W dada por la expresión

$$S = k_B \cdot \ln W \tag{4.9}$$

La Ecuación 4.9 fue establecida por Boltzman y constituye el punto de unión entre la Termodinámica y la Mecánica Estadística. La variación de Entropía en un proceso viene dada por

$$S_2 - S_1 = k_B \cdot \ln \frac{W_2}{W_1} \tag{4.10}$$

4.3. Método de Optimización Combinatoria Termodinámica

Un problema de Optimización Combinatoria se formula a partir de un espacio de configuraciones o posibles soluciones N, y una función de coste C. El problema de optimización consiste en encontrar la solución de menor coste C en el espacio soluciones N. La *Optimización Combinatoria Termodinámica* proporciona un método

para la resolución de problemas de Optimización Combinatoria (*OC*) basado en el primer y segundo principios de la Termodinámica.

En [AK89] se establece una analogía entre la optimización con Enfriamiento Simulado y la Física Estadística. De forma similar, con el objetivo de establecer una relación clara de la Termodinámica y la Física Estadística con la Optimización Combinatoria, vamos a definir en el ámbito de la *OC*, los conceptos de estado, macroestado y microestado.

En *OCT* definimos el concepto de macroestado asociado a una temperatura *T*, como una variable aleatoria X_T que toma valores en el espacio de soluciones *N* con una distribución de probabilidad

$$P\{X_T = x_i\} = p_i \qquad x_i \in N$$

de forma que

$$\sum_{i=1}^{n} p_i = 1$$

donde n es el número de posibles configuraciones en N. La forma que toman las probabilidades p_i interesa en el enfoque estadístico, puesto que a partir de estos valores se pueden determinar distintas propiedades. Así, por ejemplo, el coste C_T (equivalente a la energía en Termodinámica) de un macroestado X_T es igual a su valor medio, es decir

$$C_T = \sum_{i=1}^{n} c_i p_i \tag{4.11}$$

donde c_i es el coste de la solución x_i. De acuerdo con la definición de entropía dada por la teoría de la información [Sha48], la entropía S_T asociada a un macroestado X_T la podemos expresar como:

$$S_T = \sum_{i=1}^{n} p_i \cdot s_i$$

$$S_T = -\sum_{i=1}^{n} p_i \cdot ln(p_i) \tag{4.12}$$

Denominamos microestado a cada uno de los posibles valores x_i que puede tomar la variable X_T, junto con su probabilidad asociada p_i. Los microestados se caracterizan igualmente por un coste c_i y una entropía s_i que de acuerdo con la Ecuación 4.12 vendrá dada por

$$s_i = -ln(p_i) \tag{4.13}$$

Definimos el concepto de estado como el valor x_i de la variable X_T en un momento dado. A cada estado también le corresponde un coste c_i y una entropía s_i dada por la Ecuación 4.13.

Una vez determinada la terminología que vamos a utilizar, podemos decir que la *Optimización Combinatoria Termodinámica* consiste en dos procesos de optimización superpuestos: uno local y otro global. El primero se centra en transformaciones locales entre estados, mientras que la optimización global consiste en un recorrido a través de una serie de macroestados.

De forma general, durante el proceso de optimización local interesará que los cambios de estado ocurran alcanzando el máximo rendimiento. De acuerdo con la Termodinámica, un proceso reversible realizado entre dos estados x_A y x_B alcanza un rendimiento mayor que cualquier otro proceso realizado irreversiblemente. Por lo tanto, el primer objetivo de la *OCT* va a consistir en realizar el movimiento entre estados mediante procesos reversibles. Es decir, se exige que todas las transformaciones locales entre dos estados cumplan el segundo principio de la Termodinámica para procesos reversibles realizados a la temperatura T (Ecuación 4.14).

$$S_B - S_A = \frac{Q_R}{T} \tag{4.14}$$

Por otra parte, el primer principio de la termodinámica para sistemas que sólo intercambian calor con los alrededores y no realizan trabajo toma la forma

$$U_B - U_A = Q \tag{4.15}$$

Por lo tanto, un proceso reversible sufrido por un sistema a la temperatura T, donde únicamente se cede calor a los alrededores debe cumplir

$$U_B - U_A = T \cdot (S_B - S_A) \tag{4.16}$$

Si hacemos la equivalencia entre Energía Interna (U) y coste de un estado (c), para problemas de optimización combinatoria tenemos

$$c_B - c_A = T \cdot (s_B - s_A) \tag{4.17}$$

Por otra parte, de acuerdo con la definición de la entropía de un estado dada por la Ecuación 4.13, tenemos

$$s_B - s_A = -ln \frac{p_B}{p_A} \tag{4.18}$$

donde p_A *y* p_B son las probabilidades que corresponden a los estados x_A y x_B. Durante el proceso de búsqueda local se van a generar nuevos estados que se van a someter a un test de aceptación. Si partimos de un estado x_A, la probabilidad p_B de alcanzar el estado x_B será P_{AB}. De la misma forma, la probabilidad p_A de permanecer en el estado x_A será igual a 1- P_{AB}. Sustituyendo estos valores en la Ecuación 4.18 tenemos

$$s_B - s_A = -ln \frac{P_{AB}}{1 - P_{AB}} \tag{4.19}$$

y por tanto

$$c_B - c_A = -T \cdot ln \frac{P_{AB}}{1 - P_{AB}} \tag{4.20}$$

Despejando P_{AB} obtenemos

$$P_{AB} = \frac{1}{1 + e^{\frac{\Delta C}{T}}} \tag{4.21}$$

Por lo tanto, toda transformación local realizada sobre el sistema que se someta al test de aceptación dado por la Ecuación 4.21 cumple el primer y segundo principios definidos para procesos reversibles. La Ecuación 4.21 tiene la forma de la distribución de Fermi-Dirac. En la práctica, podemos hacer algunas aproximaciones. Consideremos el caso correspondiente a una transformación entre los estados x_A y x_B realizada en k iteraciones. Tendremos que

$$P_{AB} = \prod_{i=1}^{k} P_i \tag{4.22}$$

y como $1 - \prod_{i=1}^{\kappa} P_i \approx 1$ después de algunas iteraciones, la Ecuación 4.20 toma la forma

$$c_B - c_A = -T \cdot ln\, P_{AB} \tag{4.23}$$

de donde

$$P_{AB} = \begin{cases} 1 & \Delta C \leq 0 \\ e^{-\frac{\Delta C}{T}} & \Delta C > 0 \end{cases} \tag{4.24}$$

La probabilidad de aceptación dada por la Ecuación 4.24 corresponde a la distribución de Boltzman. En el Enfriamiento Simulado el descenso de la temperatura T se realiza de forma controlada con el fin de mantener el sistema cerca del equilibrio.

En *OCT*, para determinar la variación de la temperatura, se considera un proceso de optimización global superpuesto al de optimización local. Este proceso consiste en un recorrido *macroscópico* realizado a través de una serie de macroestados. Los extremos de ese recorrido se pueden identificar fácilmente. Para iniciar el proceso de búsqueda sobre un espacio de soluciones N, y a falta de mayor información, interesará comenzar la búsqueda macroscópica a partir de un macroestado X_{T_0} donde todas las configuraciones sean igualmente probables. Es decir

$$P\{X_T = x_i\} = \frac{1}{n} \qquad \forall\, x_i \in N$$

Para ese macroestado inicial X_{To} la entropía se puede calcular como

$$S_{T_0} = -\sum_{i=1}^{n} \frac{1}{n} \cdot \ln(\frac{1}{n})$$

de donde

$$S_{T_0} = ln(n)$$

Intuitivamente, podemos considerar el macroestado inicial como un macroestado indefinido donde todas los soluciones tienen igual probabilidad. A este macroestado le corresponde la máxima entropía. A medida que avanza el proceso de optimización no sólo debe disminuir el coste, sino también la entropía. Es decir, se debe perder incertidumbre o indefinición, y ganar certeza o información. El objetivo final del proceso de búsqueda debe consistir en alcanzar una solución óptima. Si N_{opt} es el conjunto de soluciones óptimas y n_{opt} el número de elementos de ese conjunto, el macroestado final X_{Tf} debe verificar

$$P\{X_{Tf} = x_i\} = \begin{cases} \dfrac{1}{n_{opt}} & x_i \in N_{opt} \\ 0 & x_i \notin N_{opt} \end{cases}$$

donde coste medio se corresponda con el valor óptimo

$$C_{Tf} = \sum_{i=1}^{n_{opt}} \frac{1}{n_{opt}} \cdot C_{opt} = C_{opt}$$

y cuya entropía sea mínima (máxima definición)

$$S_{Tf} = -\sum_{i=1}^{n_{opt}} \frac{1}{n_{opt}} \cdot \ln(\frac{1}{n_{opt}})$$

Para el caso de que exista una única solución óptima

$$S_{Tf} = -1 \cdot ln(1) = 0$$

Debemos destacar que el descenso en el valor de la entropía es necesario. Así, durante el proceso de optimización local se puede alcanzar el estado correspondiente a la solución óptima, pero si ese estado se visita con una entropía alta (gran incertidumbre) en pocas iteraciones se perderá.

Una vez identificados los macroestados inicial y final interesará realizar un recorrido eficiente entre ambos. Es decir, la optimización debe transcurrir entre dos macroestados recorriendo el *camino* más corto posible. De acuerdo con la Termodinámica, un proceso reversible realizado entre dos macroestados X_A y X_B alcanza un rendimiento mayor que cualquier otro proceso realizado irreversiblemente. Nuestro objetivo va a consistir, por lo tanto, en movernos desde el macroestado inicial X_{T0} al macroestado final X_{Tf} mediante un proceso reversible que cumpla el primer y segundo principios (Ecuación 4.25).

$$T = \frac{C_B - C_A}{S_B - S_A} \tag{4.25}$$

Puesto que el espacio de macroestados se puede considerar continuo, podemos aplicar la estadística de Boltzman a los cambios de macroestado de forma que se cumplirá

$$S_B - S_A = \ln(P_{AB}) \tag{4.26}$$

Además, debido a la definición probabilística de macroestado, para apreciar un cambio desde esta perspectiva, no basta con aplicar el primer y segundo principios (Ecuación

4.25) a un cambio de estado simple, sino que tendremos fijar el cumplimiento del segundo principio para toda transformación macroscópica compuesta por varias transiciones locales. Supóngase una transformación entre los macroestados X_A y X_B llevada a cabo mediante k transiciones locales. Sea ΔC_i la variación de coste correspondiente a la transformación local probada en la iteración i, T_i la temperatura a la cual se probó, y P_i la probabilidad de aceptación. En este caso la variación de entropía será

$$S_B - S_A = ln \prod_{i=1}^{k} P_i \tag{4.27}$$

por lo tanto

$$S_B - S_A = \sum_{i=1}^{\kappa} ln\ P_i \tag{4.28}$$

Por otra parte, si $M^k_{aceptados}$ es el conjunto de transformaciones aceptadas hasta la iteración k, la variación de coste producida será

$$C_B - C_A = \sum_{i \in M^k_{aceptados}} \Delta C_i \tag{4.29}$$

Sustituyendo las Ecuaciones 4.28 y 4.29 en la Ecuación 4.25, la transformación macroscópica realizada hasta la iteración k, será reversible si T cumple la Ecuación 4.30. Puesto que el valor de T dado por la Ecuación 4.30, será aplicado en la siguiente iteración, lo denominamos T_{k+1}.

$$T_{k+1} = k_A \cdot \frac{\sum_{i \in M^k_{acept}} \Delta C_i}{\sum_{i=1}^{\kappa} \ln P_i} \tag{4.30}$$

Por conveniencia, en la Ecuación 4.30 se ha introducido una constante k_A que permite controlar el compromiso tiempo/calidad. Debido a que k_A es proporcional al tiempo de optimización, su valor se puede ajustar al tiempo disponible $t_{disponible}$ por medio de la Ecuación 4.31.

$$k_A = \frac{t_{disponible}}{t_{K_A=1}} \tag{4.31}$$

donde $t_{k_A=1}$ es el tiempo de ejecución medido con k_A=1. En cuanto a la temperatura inicial, su valor dependerá de la solución de partida. Podemos distinguir dos modos de funcionamiento: normal y adaptativo. En el modo normal, para soluciones de partida generadas de forma aleatoria, la temperatura inicial se puede calcular de forma similar a [WL86] según la ecuación siguiente

$$T_o = -\frac{\overline{|\Delta C|}}{\ln(P)}$$

donde $\overline{|\Delta C|}$ es la media del valor absoluto de la variación de coste obtenida para una secuencia inicial de movimientos aleatorios, y P es una probabilidad de aceptación alta cercana a 1, de forma que inicialmente se acepten casi todos los movimientos. En el modo adaptativo, que se debe utilizar siempre que la solución de partida no se genere de forma aleatoria (p.e una solución preoptimizada), se debe tomar un valor muy bajo para la temperatura inicial. Tras unos pocos movimientos, la temperatura se adaptará a la calidad de esa solución inicial.

Otro aspecto a tener en cuenta es que, al inicio de la optimización, la temperatura dada por la Ecuación 4.30 podría tomar valores negativos. Esto ocurrirá siempre que la solución actual tenga un coste mayor que la solución de partida. Para forzar la minimización de la función de coste, esta situación se debe corregir. Por lo tanto, en este caso, así como en el caso en el que el denominador se mantenga con valor cero, la temperatura se debe igualar a la temperatura inicial.

En esta sección se ha presentado el método de *Optimización Combinatoria Termodinámica.* En este método se superponen dos procesos de optimización: uno local

y otro global. El proceso de optimización local realiza transformaciones de estado *reversibles* dentro de un macroestado definido por una temperatura T. El proceso de optimización global contempla un conjunto de transformaciones locales como una transformación *reversible* entre macroestados. A diferencia del proceso local, en el proceso global la probabilidad P_{AB} dada por la Ecuación 4.24 no se interpreta como la probabilidad de pasar de un estado a otro, sino como la probabilidad de pasar del macroestado X_A con sus valores de coste y entropía (C_A *y* S_A), a otro macroestado X_B con otro conjunto de valores (C_B *y* S_B). De esta forma, siempre que se aplica el test de aceptación, se modifica el macroestado del sistema. Por ejemplo, cuando una nueva solución no es aceptada la incertidumbre disminuye puesto que hay menos cambios por unidad de tiempo. Podemos decir que la solución está más definida, las probabilidades se concentran en menos soluciones, de forma que la entropía disminuye y el macroestado del sistema cambia.

4.4. Resultados experimentales

Se ha implementado el método *OCT* para la ubicación de circuitos en FPGAs con la función de coste *Bounding Box* (*BB*). En la figura se muestra el desarrollo de la optimización con *OCT* para el circuito seq.net con un valor de k_A=300. El eje de ordenadas representa el coste, mientras el eje abscisas indica el número de iteraciones o movimientos evaluados. En la figura 4.1 se muestran también las optimizaciones realizadas con el *ES* (herramienta VPR) y la *ON*.

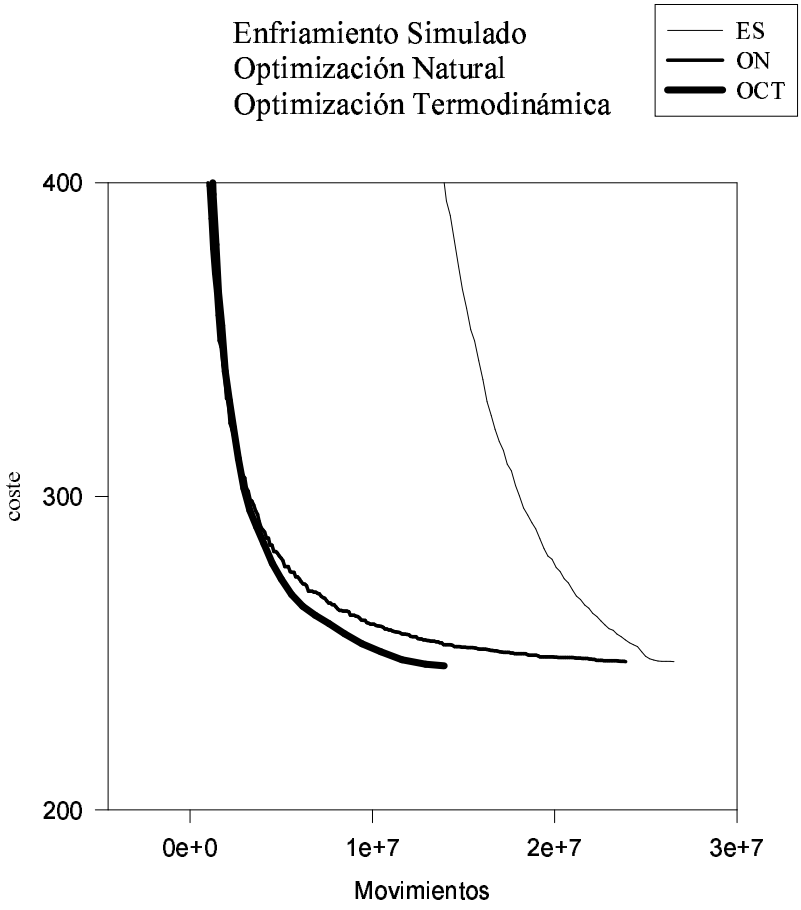


Figura 4.1 *Comparación de la evolución coste/movimientos para la optimización con ES, ON y OCT en el problema de la ubicación sobre FPGAs.*

En la figura 4.1 se observa que *OCT* reduce de forma apreciable los tiempos requeridos por la *ON* y el *ES* mientras alcanza soluciones de calidad similar.

La figura 4.2 muestra la relación entre el coste, y el decremento de temperatura representado en escala logarítmica.

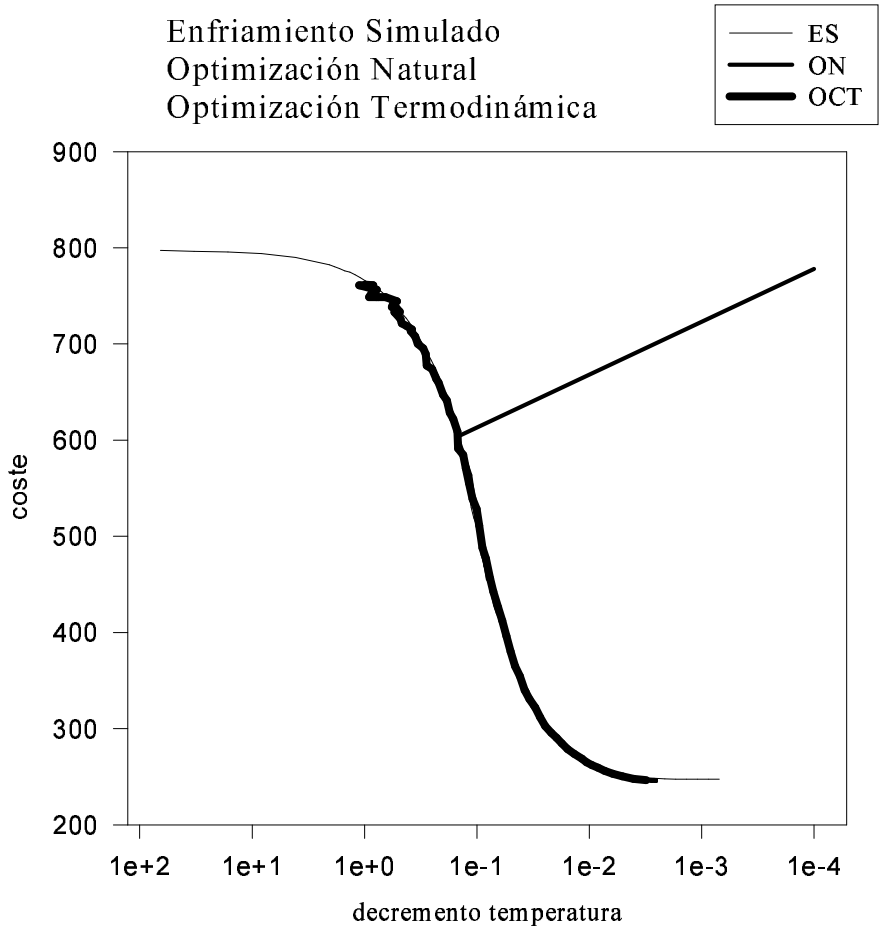


Figura 4.2 *Evolución de la relación coste/temperatura para las ubicaciones con ES, ON y OCT.*

En esta figura se observa que el método *OCT* proporciona prácticamente la misma relación *coste=f(temperatura)* que la *ON* y el *ES*. Sin embargo, como se observa en la Figura 4.3, la gráfica que relaciona la temperatura y el número de movimientos evaluados correspondiente a *OCT* tiene un comportamiento distinto respecto de la gráfica

correspondiente a la *ON*. Así, en *OCT* la temperatura desciende inicialmente de forma ligeramente más lenta, superando la pendiente de *ON* según avanza el proceso de optimización.

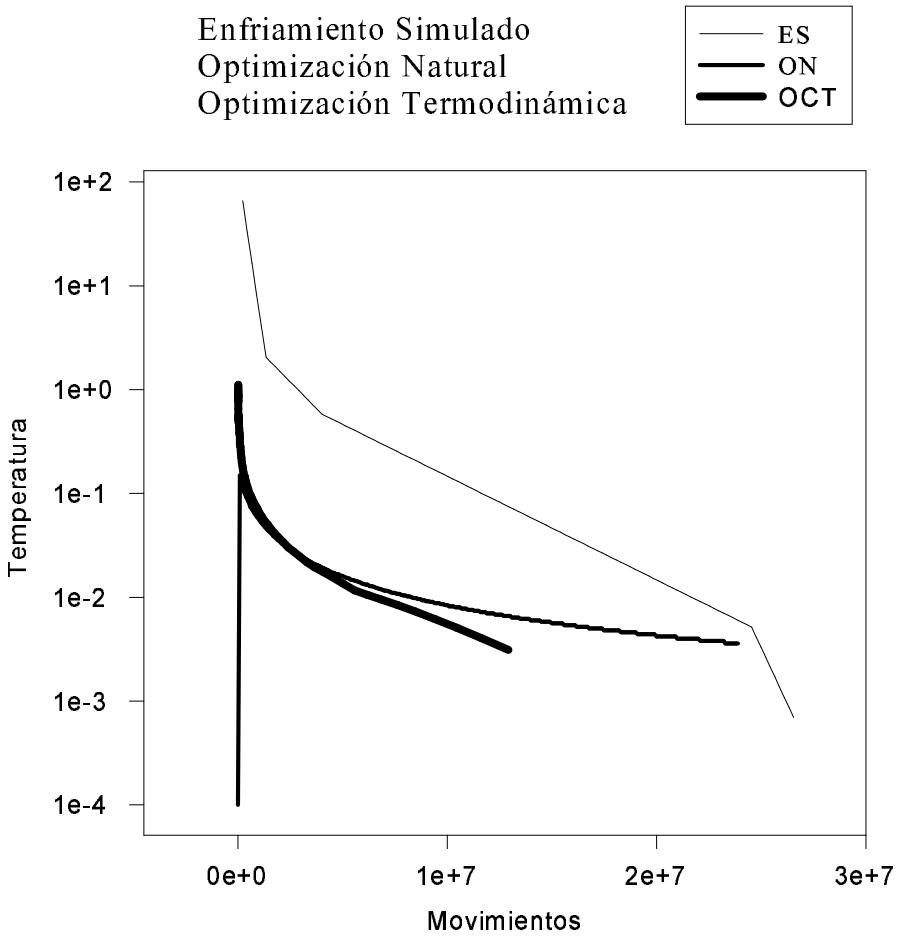


Figura 4.3 *Evolución de la relación temperatura/movimientos para las ubicaciones con ES, ON y OTC.*

Para ampliar el estudio, se ha aplicado el método *OCT* al mismo conjunto de circuitos de prueba del Capítulo 3. En la Tabla 4.1 se muestran los valores de la función de coste *BB* optimizada por *ES, ON* y *OCT,* así como el número de movimientos realizados por cada método para estos circuitos. También se muestra las mejoras relativas (calidad Q y tiempo T) de *OCT* respecto a *ES* según se definieron en el Capítulo 3. Se observa que el

método *OCT* mejora, tanto en coste como en tiempo de ejecución, los resultados proporcionados por el *ES* y la *ON*.

Tabla 4.1 ***Comparación*** *Enfriamiento Simulado (ES), Optimización Natural (ON) y la Optimización Combinatoria Termodinámica (OCT) para un conjunto de circuitos*

CIR.	Coste (Bounding Box)				Movimientos ($x10^6$)			Q	T
	Inicial	*ES*	*ON*	*OCT*	*ES*	*ON*	*OCT*	*%*	*%*
alu4	612	192	192	191	21.2	15.0	7.3	+0.2	-65
apex2	912	267	266	266	28.6	26.8	12.7	+0.2	-55
apex4	503	180	180	179	16.3	16.3	8.1	+0.3	-50
bigkey	1085	187	187	187	35.7	34.7	19.5	+0.0	-45
diffeq	670	146	145	145	22.5	16.5	11.6	+0.2	-48
dsip	895	171	171	171	28.5	36.1	22.1	+0.0	-38
ex5p	427	162	162	162	13.8	9.4	6.2	+0.0	-55
frisc	2288	516	516	516	72.4	75.9	52.3	+0.0	-27
misex3	588	188	188	188	18.8	18.0	9.7	+0.0	-48
s298	750	203	203	203	29.3	26.9	21.6	+0.0	-26
seq	804	247	246	246	26.7	23.0	12.9	+0.2	-52
tseng	420	93	93	93	15.5	9.4	6.7	+0.0	-56

4.5. Conclusión

En este capítulo, se ha presentado el método de optimización combinatoria *OCT*. *OCT* proporciona un método de enfriamiento simulado que se adapta automáticamente a distintos problemas y funciones de coste. En *OCT*, el valor de la temperatura se deduce de los principios termodinámicos. Tras cada aplicación del test de aceptación, la temperatura se actualiza de acuerdo con el primer y segundo principios de la termodinámica para procesos reversibles.

El método *OCT* se ha utilizado en la ubicación de circuitos en FPGAs, mejorando los resultados proporcionados por la herramienta *de Enfriamiento Simulado VPR*, y por el método de *Optimización Natural* presentado en el Capítulo 3. El método *OCT* preserva las propiedades adaptativas de la *ON*, mientras mejora su rendimiento.

Capítulo 5

Estimación de la congestión en la fase de ubicación:

Valores Esperados de Ocupación

5.1 Introducción

Como se ha comentado en capítulos anteriores, la fase de ubicación afecta notablemente a la distribución de las redes del circuito entre los segmentos de canal de la FPGA. Debido a que la anchura o capacidad de los segmentos de canal de la FPGA es limitada, una ubicación descuidada puede comprometer el éxito del cableado, y con ello, la implementación del circuito sobre la FPGA. Así, la congestión de los segmentos de canal de la FPGA, se debe combatir en las dos fases del diseño físico que tienen influencia directa sobre ella: las fases de ubicación y rutado. Asimismo, en la fase de ubicación se determina la distancia entre los nodos a conectar, afectando ésta a los retardos de interconexión. El capítulo anterior se centró en el desarrollo de un método de optimización combinatoria fácil de sintonizar para diferentes funciones de coste y estrategias de optimización, que además produjese resultados de calidad. En este capítulo, supuesto que disponemos de un método de optimización combinatoria fiable, afrontamos la búsqueda de una función de coste que en la etapa de ubicación haga previsiones de recursos para evitar la congestión durante el cableado. Por lo tanto, en este capítulo se analiza el problema de la congestión y de los retardos de interconexión desde el punto de vista de la fase de ubicación, explorando funciones de coste alternativas a las actuales.

Aunque en un principio, los objetivos de evitar la congestión y reducir el retardo pudieran parecer contrapuestos, un análisis detallado indica que los dos objetivos tienen un fondo común. Como veremos más adelante, la congestión global o media es igual a la longitud de cable total dividido entre el número de segmentos de canal de la FPGA. Puesto que el número de segmentos de canal de la FPGA es fijo, reducir la congestión media equivale a minimizar la longitud de cable total. Por lo tanto, ése es el primer

objetivo que debe plantearse a la hora de evitar la congestión. Por otra parte, reducir el retardo de las señales en la fase de ubicación implica minimizar la distancia entre los nodos que componen las redes del circuito, es decir, minimizar la longitud de cable total necesario. Así, la minimización de la longitud de cable total en la fase de ubicación permite alcanzar los dos objetivos de forma conjunta: reducir tanto los retardos producidos en las señales como la congestión media. Debido a los beneficios directos e indirectos que aporta la minimización del cable total, ése es el objetivo habitual de las herramientas de ubicación.

Una vez alcanzado este objetivo en el que se han minimizado los retardos de las señales en su conjunto, y se ha reducido la congestión media, se puede refinar la ubicación para alcanzar otros objetivos menos evidentes, pero igualmente importantes. Así, algunas herramientas de ubicación plantean un tratamiento específico en la fase de ubicación para las redes que componen el camino crítico. Nosotros consideramos que ese tratamiento en la fase de ubicación es poco efectivo; el cableado de las redes del circuito no se realiza siempre por el camino más directo, con el objetivo de evitar regiones congestionadas. De esta forma, el camino crítico depende en general de cómo se realice cableado final, y por lo tanto está condicionado por la herramienta de cableado que se utilice. Por esta razón, nosotros consideramos que una vez reducidos los retardos de las redes del circuito en su conjunto, es más efectivo hacer el tratamiento específico del retardo del camino crítico en la etapa de cableado mediante un esquema de prioridades, en lugar de sobrecargar la etapa de ubicación con objetivos que escapan a su influencia. Sin embargo, sí queda por resolver un problema en el que no se ha profundizado suficientemente hasta el momento: la congestión local. Ésta se produce por una distribución poco uniforme de las redes del circuito entre los segmentos de canal de la FPGA. En este capítulo se analiza el problema de la congestión local, y se proporciona un modelo probabilístico *(VEO)* para estimarla, así como una función de coste que permite evitarla.

El resto de este capítulo está organizado como sigue: en la siguiente sección se profundiza en el problema de la congestión, la forma de estimarla y medirla. En la sección 5.3 se desarrolla el modelo probabilístico *VEO* para estimar la congestión local

independientemente de la herramienta de cableado utilizada. En la sección 5.4 se presentan un conjunto de resultados experimentales que ponen a prueba este modelo. Por último, en la sección 5.5 se discute el ámbito de aplicación de este modelo probabilístico.

5.2 Congestión

Los recursos generales de interconexión de una FPGA constan de una red de segmentos de canal horizontales y verticales compuestos por cables (Figura 5.1). Al número de cables por segmento de canal se le denomina *anchura de canal* (*W*), y viene fijado por la arquitectura de la FPGA. Los recursos generales de interconexión se pueden representar por medio de un grafo G(V,E) (Figura 5.2), donde $V=\{v_1, v_2,..., v_{|V|}\}$ es el conjunto de puntos de cruce entre los canales horizontales y verticales, y $E=\{e_1, e_2,..., e_{|E|}\}$ es el conjunto de segmentos de canal que conectan los puntos de cruce. El cableado de un circuito sobre la FPGA, se realiza conectando segmentos de cable a través de unos interruptores programables situados en los puntos de cruce.

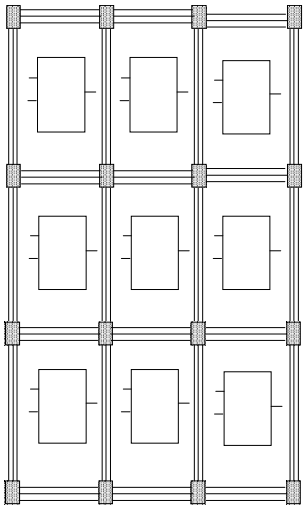


Figura 5.1 *Recursos de interconexión de la FPGA*

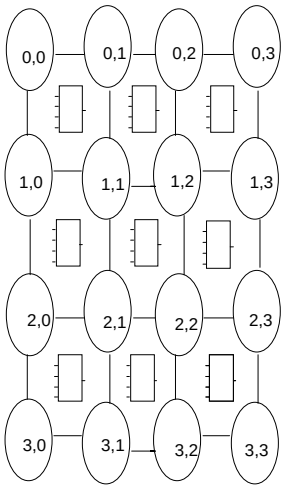


Figura 5.2 *Representación de los recursos de interconexión mediante un grafo*

5.2.1 Congestión media

La ocupación promedio ($\overline{O}$) (Ecuación 5.1) de los segmentos de canal de la FPGA producida por el cableado del circuito, se puede determinar dividiendo la longitud de cable total, entre el número de segmentos de canal de la FPGA.

$$\overline{O} = \frac{cable\ total}{segmentos\ de\ canal} \qquad (5.1)$$

Podemos definir la congestión media C_m como el cociente entre la ocupación media de los segmentos de canal y la anchura W de tales segmentos (Ecuación 5.2).

$$C_m = \frac{\overline{O}}{W} \qquad (5.2)$$

La congestión media, por lo tanto, sólo puede ser medida después de realizar el cableado. Sin embargo se pueden hacer buenas aproximaciones de la longitud de cable total, y por tanto de la congestión media, mediante la métrica denominada del *semiperímetro*. El *semiperímetro* aproxima el coste de cablear una red multiterminal por la mitad del perímetro del área que la engloba. Funciones de coste basadas en esta métrica son habitualmente utilizadas con éxito en la fase de ubicación, reduciendo los retardos en las señales y la congestión media producida por las redes.

5.2.2 Congestión local

Como se comentó en la introducción, la congestión local proviene de una distribución poco uniforme de las redes del circuito entre los segmentos de canal. Sea O_k el número de cables ocupados del segmento de canal e_k después de las fases de ubicación y rutado, y sea O el mayor de los valores O_k (Ecuación 5.3). Entonces, el

valor de O debe ser menor o igual que la anchura de canal W para que el circuito se pueda cablear.

$$O = max(O_k) \qquad para \quad k = 1,...,|E| \tag{5.3}$$

Por lo tanto, para cumplir esta restricción se debe minimizar el valor de O-W siempre que sea positivo. Al cociente entre O y W, le denominamos índice de Congestión local (C_l) (Ecuación 5.4), y su valor debe ser menor o igual que uno para que el circuito se pueda cablear.

$$C_l = \frac{O}{W} \tag{5.4}$$

En la siguiente sección se desarrolla un modelo probabilístico que permite estimar la congestión local para cualquier ubicación independientemente de la herramienta de cableado utilizada.

5.3 Valores esperados de ocupación *(VEO)*

Se define el *valor esperado de ocupación* del segmento de canal e_k (EO_k) como la *expectación* de la ocupación O_k debida a todas las conexiones del circuito, y se define el *máximo esperado* (EO) como el mayor de los EO_k. En esta sección se desarrolla el modelo probabilístico *VEO* para calcular estos valores esperados de ocupación, y con ello, estimar la congestión.

5.3.1 Cálculo de los valores esperados de ocupación

Los *VEO* de los segmentos de canal debido a todas las conexiones, se pueden calcular descomponiendo las redes multiterminal en redes de dos terminales, y suponiendo que el cableado de las redes resultantes se realiza por alguno de los posibles

caminos más cortos. Con esta suposición, sólo los segmentos de canal encerrados por un rectángulo que englobe a los nodos fuente y destino de cada red, tendrán probabilidad distinta de cero de ser utilizados en esa conexión. La idea para el cálculo de los *VEO* consiste en explorar, dentro de ese rectángulo, los posibles caminos a través de los cuales se puede rutar la red. Durante la exploración de cada camino, se asignan probabilidades de ocupación o de paso, a los segmentos de canal que lo componen. Cuando a un camino le surgen dos alternativas, éste se bifurca en dos, dividiéndose la probabilidad de paso entre los dos nuevos caminos. Por el contrario, en la unión de dos caminos las probabilidades se suman. Calculando estas probabilidades de ocupación para las redes de dos terminales que componen una red multiterminal, se puede hallar la probabilidad de ocupación de cada segmento de canal debido a esa red multiterminal. Para realizar este cálculo hay que tener en cuenta que las redes que componen una red multiterminal comparten caminos. Por lo tanto, la probabilidad de ocupación de un segmento de canal debido a una red multiterminal, se determina como la probabilidad de la unión de sucesos correspondiente a la ocupación del segmento de canal por parte de cada una de las redes de dos terminales que componen la red multiterminal. Una vez determinada la probabilidad de ocupación del segmento de canal debido a una red multiterminal, se calcula la aportación al valor esperado de ocupación del segmento de canal debido a esa red. Extendiendo este cálculo a todas las redes multiterminal, se determinan los valores esperados de ocupación para el rutado del circuito. En lo que sigue, se proporciona una descripción más formal del modelo *VEO*.

Sea $N=\{N_1, N_2,..., N_N\}$ la lista de redes a cablear, donde cada $N_i=\{N_{i1}, N_{i2},..., N_{iMi}\}$ es una red multiterminal descompuesta en redes de dos terminales, y sea G(V,E) el grafo de los recursos de interconexión de propósito general de la FPGA. El primer asunto que debemos tratar es la determinación, para cada red de dos terminales N_{ij}, de los vértices (V_{fuente}) y destino ($V_{destino}$) pertenecientes a G, a partir de los CLBs fuente y destino de N_{ij}. Es decir, determinar el vértice más cercano al CLB fuente (destino) por el cual sale (llega) la señal. Así, entre los dos vértices más cercanos a la salida del CLB fuente, y entre los dos vértices más cercanos a la entrada del CLB destino, se eligen como fuente y destino para realizar la conexión, aquel par de vértices que estén más

cercanos entre ellos. Por ejemplo, en la Figura 5.3 el vértice fuente (determinado a partir del CLB fuente (S)) es el nodo (1,1), y el vértice destino (determinado a partir del CLB destino (T)) es el nodo (2,4).

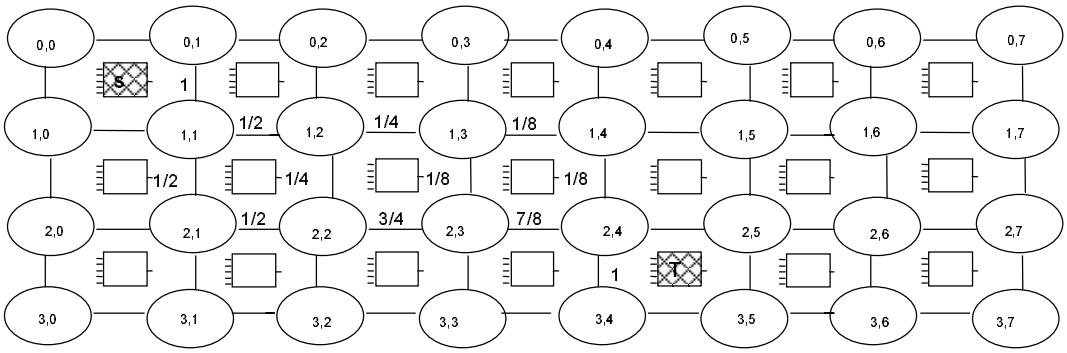


Figura 5.3 *Ejemplo de cálculo de la probabilidad de ocupación debido a una red de dos terminales*

Una vez fijados los vértices fuente y destino de la red sobre G, el cálculo de los valores esperados de ocupación se realiza a partir de la definición de unas serie de variables aleatorias. Este cálculo lo podemos dividir en tres etapas:

a) Probabilidad de que el arco e_k sea utilizado para cablear la **conexión de dos terminales** N_{ij}.
b) Probabilidad de que el arco e_k sea utilizado para cablear la **red multiterminal** N_i .
c) Valor Esperado de Ocupación del arco e_k debido a todas las conexiones.

a) Probabilidad de que un arco e_k sea utilizado para cablear la conexión de dos terminales N_{ij}

Sea $x_{ij}(e_k)$ (i=1..N, j=1...M_i, k=1..|E|) una variable aleatoria asociada al arco e_k y a la red N_{ij} que toma los valores:

$$x_{ij}(e_k) = \begin{cases} 1 \;\; si\ el\ arco\ e_k\ es\ utilizado\ para\ cablear\ N_{ij} \\ 0 \quad si\ no\ es\ utilizado \end{cases}$$

Dada una red N_{ij} entre un CLB fuente y otro destino, el segmento de canal e_k en el cual fluye la salida del CLB fuente, tiene una probabilidad igual a uno de ser utilizado para cablear la red N_{ij}. De la misma forma, el segmento de canal más cercano a la entrada del CLB destino, tiene una probabilidad igual a uno para ser utilizado para cablear la red N_{ij}. Además, todos los arcos e_k encerrados en el rectángulo que delimitan los vértices fuente (V_{fuente}) y destino ($V_{destino}$), tendrán una probabilidad distinta de cero de ser utilizados en el cableado de la red N_{ij}. Estas probabilidades pueden ser obtenidas de forma recursiva a partir de las calculadas anteriormente. Para este cálculo, se supone un sentido de recorrido de los segmentos de canal marcado por la dirección fuente-destino. La probabilidad P_{ijk} de que el arco e_k sea utilizado para cablear la red de dos terminales N_{ij} se puede calcular teniendo en cuenta las dos propiedades siguientes:

1) La probabilidad de que la conexión se realice a través de un vértice v_i, será igual a la suma de las probabilidades correspondientes a los arcos e_k que con dirección hacia destino convergen en ese vértice.
2) La probabilidad de un vértice v_i se reparte equitativamente entre los arcos e_k que salen de ese vértice en dirección al destino.

En la Figura 5.3 podemos apreciar algunos ejemplos de la aplicación de estas propiedades. Para la primera propiedad podemos fijarnos, por ejemplo, en el vértice (2,4). La probabilidad de que la conexión entre el CLB fuente S y el CLB destino T, se realice por pasando por el vértice (2,4) ($P_{(2,4)}$), será igual a la suma de las probabilidades de los segmentos o arcos que convergen en el vértice (2,4) con dirección al destino (segmentos ((2,3),(2,4)) y ((1,4),(2,4))). Es decir:

$$P_{(2,4)} = P_{((2,3),(2,4))} + P_{((1,4),(2,4))} = 7/8 + 1/8 = 1$$

Como ejemplo de la aplicación de la segunda propiedad podemos fijarnos en el vértice (1,1) de la Figura 5.3. Aplicando la primera propiedad, la probabilidad de que la conexión entre el CLB fuente S y el CLB destino T, se realice pasando por el vértice (1,1) es $P_{(1,1)} = P_{((0,1),(1,1))} = 1$. Ahora bien, esta probabilidad se reparte equitativamente entre los arcos o segmentos ((1,1),(1,2)) y ((1,1),(2,1)) que partiendo del vértice (1,1) se dirigen hacia el destino. De esta forma tenemos

$$P_{((1,1),(1,2))} = P_{(1,1)}/2$$
$$P_{((1,1),(2,1))} = P_{(1,1)}/2$$

Por lo tanto, aplicando las propiedades 1) y 2) podemos calcular, por medio de la Ecuación 5.5, las probabilidades de ocupación P_{ijk} (del arco e_k debido a red N_{ij}), en función de los valores de estas mismas probabilidades calculadas para los arcos que preceden al arco e_k en el recorrido realizado desde el vértice fuente al vértice destino. En la Ecuación 5.5, la probabilidad de que el arco k que une los vértices v_p y v_l sea utilizado para conectar la red N_{ij} que une los vértices v_{fuente} y $v_{destino}$ ($P\{x_{ij}(e_k(v_p,v_l))=1\}$), es igual a la suma (primera propiedad) de las probabilidades de los arcos que, con origen en un vértice más cercano al v_{fuente} que al $v_{destino}$ (conjunto I), desembocan en v_p (arcos ya recorridos que desembocan en el vértice origen del arco sobre el cual se quiere hallar la probabilidad), dividido (segunda propiedad) por el número de caminos alternativos (número de elementos del conjunto OU) abiertos a partir del vértice v_p con dirección al destino.

$$P_{ijk} \equiv P\{x_{ij}(e_k(v_p, v_l)) = 1\} = \sum_{v_m \in I} \frac{P\{x_{ij}(e(v_m, v_p)) = 1\}}{|OU|} \qquad (5.5)$$

con $I = \{v_m\} \subset V / d(v_m, v_p) = 1 \wedge d(v_m, v_{fuente}) < d(v_p, v_{fuente})$

$OU = \{v_l\} \subset V / d(v_p\, v_l) = 1 \wedge d(v_l, v_{destino}) < d(v_p, v_{destino})$

donde $d(v_i, v_j)$ *es la distancia Manhattan entre los vértices* v_i, v_j.

El cálculo de las probabilidades P_{ijk} es la base del modelo *VEO*. Así, los demás cálculos se realizarán a partir de las probabilidades P_{ijk}. Se pueden idear diversos métodos para el cálculo de las probabilidades P_{ijk}. Inicialmente se desarrolló un método mediante el cual se recorren todos los posibles caminos más cortos entre el vértice fuente y el vértice destino de la red. Durante estos recorridos se calculan las probabilidades de ocupación P_{ijk} de cada arco por el que pasan. Se parte con una probabilidad de ocupación igual a uno para el arco conectado a la salida del CLB, y en cada bifurcación de caminos la probabilidad se divide en dos partes equitativas (segunda propiedad). Así, teniendo en cuenta que todos los posibles recorridos son mútuamente excluyentes (dos recorridos pueden pasar por el mismo segmento de canal, pero sólo un recorrido se llevará a la práctica), las probabilidades de ocupación de un segmento de canal e_k proporcionadas por cada posible recorrido se pueden sumar (primera propiedad).

Este primer método de cálculo de las probabilidades de ocupación resultó demasiado lento. Un examen detallado del método, nos da las razones de su lentitud. Sean *m* y *n* las dos componentes de la distancia rectilínea entre un vértice fuente y otro destino. El número de caminos entre ambos nodos viene dado por la expresión:

$$\binom{n+m}{n} = \frac{(n+m)!}{n!m!}$$

Podemos hacer alguna aproximación cuando m=n. En este caso la expresión se convierte en :

$$\frac{(2n)!}{n!n!}$$

Aplicando la aproximación de Stirling ($n! \approx \sqrt{2\pi} \cdot n^{n+1/2} \cdot e^{-n}$) y para n grande tenemos:

$$Número\ de\ caminos \approx \frac{1}{\sqrt{\pi \cdot n}} \cdot 2^{2n}$$

Es decir, el número de caminos crece exponencialmente con la distancia entre los vértices. Por lo tanto, para distancias grandes, el recorrido de todos los caminos posibles entre el nodo fuente y el destino para el cálculo de las probabilidades se hace inviable.

Para mejorar la velocidad de cálculo de las probabilidades de ocupación, se desarrollo un segundo algoritmo *(Cálculo de la probabilidad de ocupación)* (Figura 5.4). Para facilitar los cálculos, en este método se determinan también las probabilidades de paso a través de todos los vértices encerrados en el rectángulo que delimitan los vértices fuente y destino. Así, a cada vértice v_i se le caracteriza por sus coordenadas en la malla $c(v_i)$, y una probabilidad de ocupación $P(v_i)$. De la misma forma, a cada arco $e_k \in E$, se le asigna un peso $P(e_k)$ que acumulará el valor EO_k durante su cómputo. En este método, en lugar de recorrer todos los posibles caminos, se expande un *frente de onda* desde el nodo fuente hacia el nodo destino. La expresión *frente de onda* hace referencia al orden en que se evalúan las probabilidades de los vértices, que se establece de acuerdo con la distancia de cada vértice al vértice fuente. Durante la expansión de este frente de onda, se calculan las probabilidades de los segmentos de canal que recorre basándose en las dos propiedades descritas anteriormente para su cálculo. La Figura 5.3 muestra un ejemplo de este cálculo.

```
Cálculo de la probabilidad de ocupación(v_FUENTE : VERTICE; v_DESTINO : VERTICE )
    LIST:VERTICES;
    LIST_TEMP:VERTICES;
Begin
   P(v_FUENTE)=1
   LIST=v_FUENTE
   While v_i ∈ LIST ≠ v_DESTINO   loop
      For each v_i ∈ LIST    loop
         Sea W ⊂ V / w_k  ∈ W  if distancia(v_i , w_k ) = 1   and
                        Distancia( w_k , v_DESTINO) <distancia(v_i, v_DESTINO)
         Sea N=/W/
         for each  w_k  ∈ W  loop
            insertar w_k  en LIST_TEMP if w_k ∉ LIST_TEMP
            P(e(v_i ,w_k))=P(v_i)/N
            P(w_k)=P(w_k)+P(v_i )/N
         end loop
      end loop
      LIST=LIST_TEMP
   End loop
end Cálculo de la probabilidad de ocupación
```

Figura 5.4 *Algoritmo para el cálculo de las probabilidades de ocupación*

b) Probabilidad de que el arco e_k sea utilizado para cablear la red multiterminal N_i

Sea x_i (e_k) (i=1..N,k=1..|E|) una variable aleatoria asignada al arco e_k que toma los valores:

$$x_i(e_k) = \begin{cases} 1 & si\ el\ arco\ e_k\ es\ utilizado\ para\ cablear\ la\ red\ multiterminal\ N_i \\ 0 & si\ no\ es\ utilizado \end{cases}$$

Para tener en cuenta los caminos comunes de una red multiterminal, el suceso $\{x_i(e_k)=1\}$ se puede definir como la unión de sucesos dados por la Ecuación (5.6).

$$\{x_i(e_k) = 1\} = \bigcup_{j=1}^{M_i} \{x_{ij}(e_k) = 1\} \qquad (5.6)$$

La probabilidad de que ocurra este suceso es igual a una suma de muchos términos. Por ejemplo, supongamos que una red multiterminal N_i está compuesta por dos redes de dos terminales (N_{i1}, N_{i2}). La probabilidad de ocupación del segmento de canal e_k debido a la red multiterminal N_i será igual a la suma de los siguientes términos:

a) Probabilidad de que el segmento de canal sea utilizado para rutar N_{i1} (P_{i1k}) y para rutar N_{i2} (P_{i2k}).

$$t_1 = P_{i1k} \cdot P_{i2k}$$

b) Probabilidad de que el segmento de canal sea utilizado para rutar N_{i1} (P_{i1k}) y no para rutar N_{i2} ($1-P_{i2k}$).

$$t_2 = P_{i1k} \cdot (1-P_{i2k})$$

c) Probabilidad de que el segmento de canal no sea utilizado para rutar N_{i1} $(1-P_{i1k})$ y sí para rutar N_{i2} (P_{i2k}).

$$t_3=(1-P_{i1k})\cdot(P_{i2k})$$

De forma general, el número de términos crece como 2^n-1 para una red multiterminal compuesta por n redes de dos terminales. Para hacer el cálculo más sencillo, hallaremos la probabilidad del suceso contrario. Así, en el ejemplo anterior, el suceso contrario consiste en que ninguna de las dos redes que componen la red multiterminal pasen por el segmento de canal. Es decir, la probabilidad del suceso contrario se compone de un sólo término y será igual a:

a) Probabilidad de que el segmento de canal no sea utilizado para rutar N_{i2} y no sea utilizado para rutar N_{i1}.

$$t_1=(1-P_{i1k})\cdot(1-P_{i2k})$$

Para el caso general de la red multiterminal N_i compuesta por M_i redes de dos terminales, la probabilidad de que el arco e_k sea utilizado para cablear la red multiterminal será:

$$P_{ik} \equiv P\{x_i(e_k) = 1\} = 1 - \prod_{j=1}^{M_i} \overline{P_{ijk}} \qquad (5.7)$$

donde

$$\overline{P_{ijk}} = 1 - P_{ijk} \qquad (5.8)$$

Una vez calculada la probabilidad de ocupación (P_{ik}) del segmento de canal e_k debida a una red multiterminal N_i, el cálculo del valor esperado de la variable aleatoria x_i es directo, pues coincide con esta probabilidad. Así, el valor esperado de x_i es:

$$E(x_i(e_k)) = 1 \cdot P\{x_i(e_k) = 1\} + 0 \cdot P\{x_i(e_k) = 0\}$$

entonces

$$E_{ik} \equiv E(x_i(e_k)) = P\{x_i(e_k) = 1\} \equiv P_{ik} \qquad (5.9)$$

c) Valor Esperado de Ocupación del arco e_k debido a todas las conexiones

Se define una nueva variable aleatoria $X(e_k)$ como el número de cables utilizados del arco e_k debido a todas las redes (Ecuación 5.10).

$$X(e_k) = \sum_{i=1}^{N} x_i(e_k) \qquad (5.10)$$

Puesto que $X(e_k)$ es una suma de variables aleatorias, su valor esperado será igual a la suma de los valores esperados de las variables que componen la suma [Fel68]. Así, tenemos

$$EO_k \equiv E(X(e_k)) = \sum_{i=1}^{N} E_{ik} \qquad (5.11)$$

de donde podemos obtener el máximo de los valores esperados de ocupación

$$EO = max(EO_k) \qquad (5.12)$$

5.3.2 Función de coste basada en los *Valores Esperados de Ocupación*

Como hemos visto, la congestión local se puede estimar a través de los valores esperados de ocupación. Si la capacidad de cálculo, y el tiempo de cómputo disponible lo permiten, estos valores se puede calcular para cada ubicación con el fin de determinar el índice de congestión local, y de esta forma evitarla mediante algún método de optimización combinatoria. El objetivo de esta optimización deberá consistir en

minimizar el *máximo de ocupación esperada* siempre que esté por encima del valor umbral correspondiente a la anchura de canal. Una forma eficiente de reducir ese máximo es minimizando la función de coste que denominamos *VEO* (Ecuación 5.13). Es decir, minimizando la suma de los cuadrados de los excesos de los valores esperados de ocupación sobre la anchura de canal.

$$VEO = \sum_{\forall e_k \in E / EO_k > W} (W - EO_k)^2 \tag{5.13}$$

5.4 Resultados experimentales

Para evaluar experimentalmente la fiabilidad del modelo probabilístico *VEO*, se implementaron las fases de ubicación y rutado para un conjunto de 15 circuitos de prueba generados de forma aleatoria sobre una FPGA de 10x10. Las características de los circuitos como son el número de bloques lógicos *(BL)* y el número de redes dividido por *BL (k)* se muestran en la Tabla 5.1. La fase de optimización de la ubicación fue guiada por la función de coste *VEO* descrita por la Ecuación 5.13. Una vez terminada esta fase, el valor *EO* se conservó para compararlo con la ocupación real producida después de realizar el cableado global *(O)*. Este cableado se llevo a cabo descomponiendo las redes multiterminal en redes de dos terminales, y aplicando el algoritmo de "*El camino más corto*" a todas ellas. La Tabla 5.1 muestra los valores del *máximo de ocupación esperada* (EO) y del *máximo de ocupación real* (O) para los distintos circuitos.

En la Tabla 5.1 se puede observar que en casi todos los casos, el valor del máximo de ocupación real medido después del cableado coincide con la aproximación entera, al alza o a la baja, del valor esperado de ocupación estimado en la fase de ubicación. Se puede decir, por lo tanto, que estos resultados proporcionan una prueba de la fiabilidad del modelo *VEO*.

Tabla 5.1 *Máximo de ocupación esperada (EO) (estimada en la fase de ubicación) comparada con el Máximo de ocupación real (O) (medida después del cableado)*

CIRC.	*O*	*EO*	*BL*	*K*
n103	6	5.12	100	1.01
n103a	5	5.50	100	1.02
n103b	5	5.00	100	1.00
n104	7	7.25	100	1.66
n104a	7	7.00	100	1.52
n104b	6	6.78	100	1.45
n105	10	8.87	100	1.91
n105a	8	7.87	100	1.72
n105b	10	9.21	100	2.18
n106	11	10.98	100	2.60
n106a	9	9.28	100	2.37
n106b	10	10.00	100	2.29

5.5 Revisión crítica de la aplicabilidad del modelo y definición de nuevos objetivos

La bondad y el ámbito de aplicación del método de ubicación basado en el modelo *VEO*, se puede determinar a partir del siguiente análisis; existen dos tipos de herramientas de cableado diferenciadas en función de cómo realizan el tratamiento de las redes multiterminal: herramientas que descomponen las redes multiterminal en redes de dos terminales, y herramientas que trabajan directamente con las redes multiterminal. Las primeras realizan un cableado rápido, mientras que las segundas son

considerablemente más lentas puesto que su objetivo es, en último término, resolver el problema del *Árbol de Steiner Mínimo Rectilíneo* para cada red, el cual es un problema NP-completo. Sin embargo, estas últimas herramientas ahorran recursos de cableado buscando caminos comunes para los nodos de una misma red. El modelo *VEO* es válido para ambas aproximaciones ya que tiene en cuenta, probabilísticamente hablando, los caminos comunes de las redes multiterminal. Por lo tanto, la función de coste basada en el modelo *VEO* puede proporcionar buenas ubicaciones (en media) para ambos tipos de herramientas de cableado. Si bien esta característica de universalidad es deseable en muchas ocasiones, aquí se va a convertir en el principal escollo para alcanzar valores óptimos en la descongestión. La falta de definición en los criterios de rutado, hace que las zonas o segmentos de canal con probabilidad distinta de cero de ser utilizadas para rutar cada red estén muy difuminadas. Esa gran incertidumbre en cuanto al camino que seguirá cada red, impide profundizar en la optimización de las soluciones. Así, tras un análisis de los resultados que generaba la función de coste basada en el modelo *VEO*, se observó que no proporcionaba las mejores ubicaciones posibles para las mejores herramientas de cableado, es decir, aquellas que trabajan con redes multiterminal. Se consideró conveniente, por lo tanto, modificar el modelo *VEO* para que contemplase una segunda suposición: "la herramienta de cableado trabajaría directamente con redes multiterminal". Es decir, se asume que la herramienta de cableado busca caminos comunes para la interconexión de nodos pertenecientes a una misma red. Como se comenta en el siguiente capítulo, el desarrollo de este nuevo modelo fue el siguiente objetivo del trabajo de investigación.

Capítulo 6

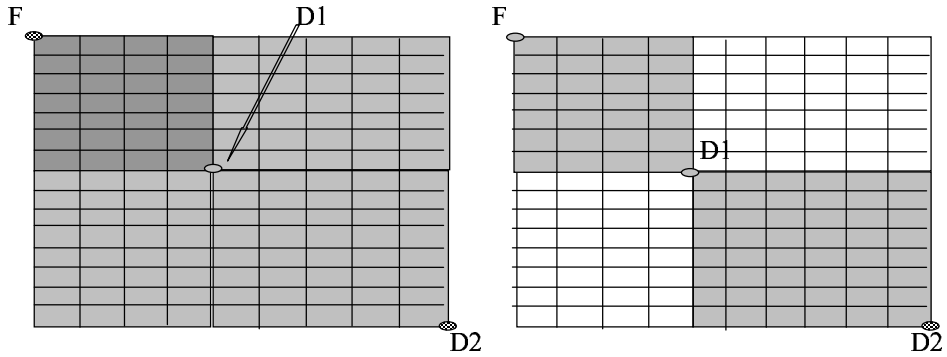
Medida de la congestión en la fase de ubicación:

Regiones de Steiner Rectilíneas

6.1 Introducción

En el capítulo anterior se desarrolló el modelo probabilístico *VEO* para estimar, en la fase de ubicación, la ocupación de los segmentos de canal de la FPGA que se producirá durante el cableado del circuito. El modelo *VEO* proporciona estimaciones fiables de la ocupación de los segmentos de canal. Además, tiene la virtud de ser independiente de la herramienta de cableado utilizada para rutar el circuito, puesto que sólo se exigió un requisito que es objetivo común de todas las herramientas: el cableado entre un nodo fuente y un nodo destino se realiza por el camino más corto siempre que sea posible. Sin embargo, esa polivalencia es precisamente el gran inconveniente de este modelo cuando se trata de alcanzar valores óptimos en la descongestión, debido a la indefinición en los criterios de rutado. Para salvar las dificultades señaladas al final del capítulo anterior, se consideró que podría ser beneficioso particularizar el modelo para un tipo de herramientas de cableado: aquéllas diseñadas para trabajar con redes multiterminal. De esta forma, se podría conseguir un doble objetivo: por una parte, optimizar la ubicación para las herramientas de cableado más eficientes. Por otro lado, se eliminaba parte de la indefinición en cuanto a los futuros caminos que tomarán las redes, permitiendo de esta forma profundizar en la optimización. La Figura 6.1 plantea el problema del cableado de una red multiterminal compuesta por un nodo fuente (F) y dos nodos destinos (D1)(D2). En la Figura 6.1a no se hacen suposiciones en cuanto a la herramienta de cableado que se utilizará, de forma que la *"nube"* de probabilidades de ocupación (zonas sombreadas) se extiende en toda el área que corresponde a caminos mínimos entre el nodo fuente y los destinos. En la Figura 6.1b, al definir el tipo de cableado como cableado multiterminal, la "nube" de probabilidades se estrecha, proporcionando

mayores posibilidades de optimización al estar definidos, de forma más precisa, los posibles caminos asociados a cada una de las redes.



a) *Cableado indefinido* b) *Cableado multiterminal*

Figura 6.1 *Las zonas sombreadas corresponden a segmentos de canal con alguna probabilidad de que la red que une la fuente F con los destinos D1 y D2 pase por ellos. En la figura a) no se establece ningún criterio de rutado, excepto que éste se realice por el camino más corto. En esta figura podemos distinguir dos sombreados distintos. En el más oscuro, el cálculo de probabilidades corresponde a una unión de sucesos (para cada uno de estos segmentos de canal existe una probabilidad distinta de cero de que alguna de las redes F-D1 o F-D2, o las dos, se ruten a través de él). El sombreado más claro corresponde a segmentos de canal con probabilidad de que F-D2 se rute a través de ellos. La figura b) corresponde a un cableado multiterminal. Puesto que el cableado de las redes F-D1 y F-D2 se realiza conjuntamente, una vez alcanzado el puntoD1, se parte de éste para alcanzar el punto D2. En la figura se observa que la nube de probabilidades se estrecha respecto al caso anterior.*

Tras este análisis, nuestro siguiente objetivo se centró inicialmente en particularizar el modelo *VEO* para herramientas de cableado diseñadas para trabajar con redes multiterminal. El estudio y desarrollo de este nuevo modelo, nos llevó de forma natural a la concepción de un método rápido de cableado de redes multiterminal, es decir, una aproximación polinómica ($o(n^2)$) al *Árbol de Steiner Mínimo Rectilíneo*. A este nuevo método le denominamos "***Regiones de Steiner Rectilíneas"*** *(RSR)* debido a su forma de trabajo basado en la utilización temporal de regiones rectilíneas indeterminadas. Este descubrimiento reorientó nuestros objetivos. Por una parte, *RSR* es un algoritmo de cableado suficientemente rápido como para poder ser utilizado como medida, en lugar de estimación, de la congestión durante la etapa de ubicación. Con esta nueva orientación, la ubicación se optimiza para una herramienta de cableado concreta. Así, se

elimina la incertidumbre en el cableado permitiendo una mayor focalización de la optimización hacia el objetivo marcado, reducir la congestión. Por otra parte, *RSR* produce aproximaciones de calidad del *Árbol de Steiner Mínimo Rectilíneo (ASMR),* siendo ese el objetivo principal de las herramientas de cableado. Por lo tanto, se puede utilizar, con ligeras modificaciones, como herramienta de cableado en sí misma. En la siguiente sección se presenta el algoritmo *RSR* y se compara con las mejores aproximaciones al *ASMR*. En la sección 6.3 se describe *U-RSR*, una estrategia de optimización en etapas para reducir la congestión local en la fase de ubicación. En la sección 6.4 se presenta el algoritmo *CG-RSR* de cableado global basado en las regiones *RSR*, y en la 6.5 se comparan los resultados producidos por la estrategia conjunta de ubicación y cableado *UCG-RSR*, con los obtenidos por la herramienta *VPR* descrita en el capítulo 2. Finalmente, en la sección 6.6 se extraen algunas conclusiones.

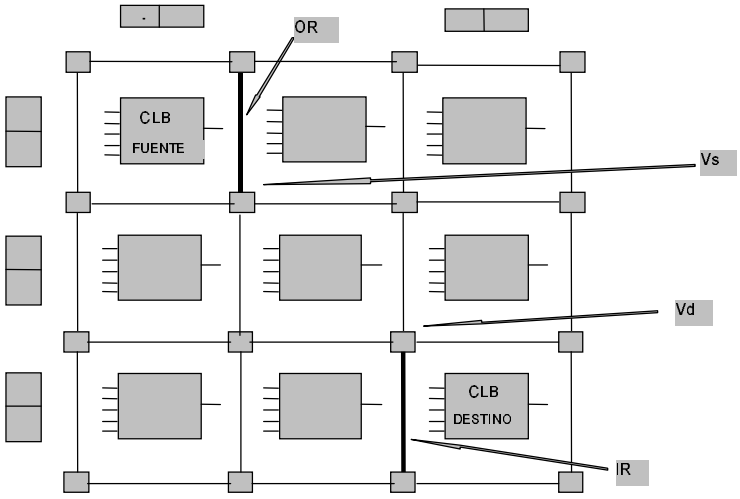


Figura 6.2 *Regiones de entrada (IR), Regiones de salida (OR), vértice fuente (v_s), vértice destino (v_d)*

6.2 Regiones de Steiner Rectilíneas

En esta sección se describe el algoritmo *Regiones de Steiner Rectilíneas* (Figura 6.3), el cual es una nueva aproximación a la solución del problema del *Árbol de Steiner Mínimo Rectilíneo* [LBH76]. Sea G(V, E) el grafo de los recursos de interconexión de

una FPGA como se describió en el capítulo anterior, y sea $N=\{M_1, M_2,..., M_n\}$ una red multiterminal descompuesta en redes de dos terminales. En la Figura 6.3(a) se introduce la terminología utilizada para describir el algoritmo. El algoritmo *RSR* (Figura 6.3(b)) acepta como entrada la ubicación de los vértices que componen la red multiterminal, y construye una lista de regiones ($\mathcal{R}$) a través de las cuales la red se puede cablear. El algoritmo comienza ordenado los vértices destino según su distancia desde el vértice fuente. Para cada uno de estos vértices destino (v_{di}), tomados en orden, determina su *Camino_común* (Figura 6.3(a)) con el camino ya rutado $\mathcal{R}$, y crea nuevas regiones para las partes del camino que no son comunes. Aunque esta primera parte del algoritmo produce algunas regiones de rutado rectangulares, y por tanto con varios caminos alternativos de la misma longitud, el camino final se puede seleccionar de forma que se evite la congestión. Como los recursos de cableado centrales de la FPGA son los más demandados, se eligen entre los distintos caminos alternativos, aquél que pasa más lejos del centro de la FPGA. Esto se lleva a cabo con la función *Elige_camino*. La Figura 6.4 muestra un diagrama simplificado correspondiente a un ejemplo de construcción de las *Regiones de Steiner Rectilíneas* desde un vértice fuente a un vértice destino. En las Figuras 6.4(a) y 6.4(b) dos vértices destino son cableados con regiones rectangulares indeterminadas. En la Figura 6.4(c) una de las regiones indeterminadas se resuelve en favor de una alternativa durante el rutado del tercer vértice destino. En la Figura 6.4(d) el cuarto vértice destino se ruta con una nueva región rectangular indeterminada. Esta región se resuelve en favor de una de sus alternativas durante el cableado del quinto vértice destino (Figura 6.4(e)). Las regiones rectangulares que permanecen, se resuelven en favor del camino más alejado del centro de la FPGA para evitar la congestión esa zona (Figura 6.4(f)). La región salida *(OR)*, que se ha omitido en el diagrama por simplicidad, se inserta al principio en la lista de regiones estableciéndose, de esta forma, la conexión con la salida del CLB fuente. En cuanto a las regiones de entrada *(IR_i)*, también omitidas en el diagrama, su inserción se realiza tras ser alcanzado el vértice destino, estableciéndose así la conexión con el CLB destino.

Definiciones asociadas al algoritmo RSR
Distancia *Distancia Manhattan o rectilínea* ***Región*** *$R \equiv R(v_{fuente}, v_{destino})$: Conjunto de vértices de G que pertenecen al perímetro del rectángulo que delimita los vértices fuente y destino de una conexión de dos terminales. A cada región se le asocia un número de red que identifica la red multiterminal que la ha generado.* ***Tamaño de una región*** *$s(R) = distancia(v_{FUENTE}, v_{DESTINO})$* ***Vértice más cercano entre vértice y región*** *$v_{cl} \equiv Vértice\_más\_cercano(v_k, R)$: $v_i \in R$ /distancia(v_k , v_i) es mínima.* ***Distancia vértice-región*** *$Distancia(v_k, R) = distancia(v_k, v_{cl})$* ***Lista de regiones*** *Conjunto de regiones: $\Re = \{R_1, ..., R_L\}$* ***Distancia vértice-lista regiones*** *$Distancia(v_k, \Re) = min(Distancias(v_k, R_j))$ con $R_j \in \Re$* ***Camino común región-vértice*** *Camino_ común (R, v_k): Operador que devuelve la lista de regiones siguiente* *$\Re = \{R(v_{FUENTE}, v_{cl}), R(v_{cl}, v_{DESTINO}), R(v_{cl}, v_k)\}$* ***Vértice más cercano entre un vértice y una lista de regiones*** *$v_{cl} \equiv Vértice\_más\_cercano(v_k, \Re)$: $v_{clj} \in R_j \subseteq \Re$ / distancia(v_k , v_{clj}) es mínima.* ***Región más cercana entre un vértice y una lista de regiones*** *$Región\_más\_cercana(v_k, \Re)$: $R_i \subseteq \Re / v_{cl} \in R_i$* ***Camino_común lista_regiones-vértice*** *Camino_común($\Re$, v_k): Operador que sustituye la región* *R=Región_más_cercana(v_k, $\Re$) por la sublista proporcionada por* *Camino_común(R , v_k).* ***Región de Salida (OR)*** *Par de vértices que soportan el arco de salida del CLB fuente (Figura 6.2). Como estas regiones soportan la salida de los CLBs tendrán máxima prioridad cuando sean cableadas.* ***Región de Entrada (IR_i)*** *Par de vértices que soportan el arco de entrada al CLB destino (Figura 6.2). Como estas regiones soportan la entrada a los CLBs tendrán también máxima prioridad cuando sean cableadas.*

Figura 6.3(a) *Definiciones para el algoritmo RSR*

RSR(net N) return lista de regiones
$\Re$: lista de regiones *begin* *Sea v_s el vértice fuente de la red N* *Sea T={ v_{di} } , i=1..n el conjunto de vertices destino de la red N* *Ordenar T de acuerdo con la distancia de sus elementos a v_s* *Sea OR la región de salida de la red N* *Insertar OR en $\Re$* *for each $v_{di} \in T$* *loop* *Camino común ($\Re$, v_{di})* *Insertar IR_i de M_i en $\Re$* *end loop* *Elige camino ($\Re$)* *return $\Re$* *end RSR*

Figura 6.3 (b) *Algoritmo RSR*

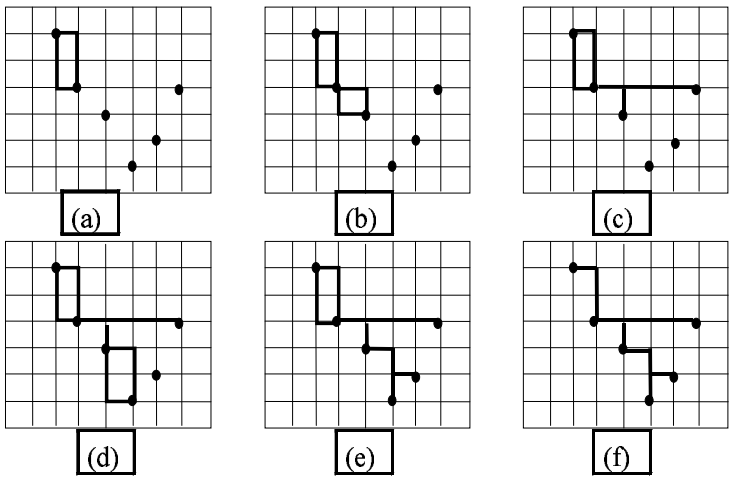


Figura 6.4 *Ejemplo de construcción de las RSR para una red multiterminal*

Si $|\mathcal{R}|$ es el número de regiones encontradas para la red N, la complejidad temporal de *RSR* será O($|\mathcal{R}|$n). El operador *Camino_común* produce una o dos regiones nuevas cada vez que se aplica, dependiendo de si el vértice más cercano es un punto terminal o de Steiner respectivamente. Entonces se cumple $|\mathcal{R}| \leq 2n$ y por lo tanto la complejidad del algoritmo RSR será $O(n^2)$.

La Tabla 6.1 muestra la longitud y el tiempo de cómputo de las aproximaciones al *Árbol de Steiner Mínimo Rectilíneo* proporcionadas por el algoritmo RSR, y por la herramienta [Rob] que implementa el algoritmo BI1S (Capítulo 2), para una serie de instancias del problema generadas de forma aleatoria por la herramienta BI1S. En la Tabla 6.1, también aparece una comparación de tiempo y calidad entre ambas aproximaciones. La calidad se mide por el porcentaje de mejora (M) respecto al *Árbol de Expansión Mínima* (AEM) según se describió en el Capítulo 2. La velocidad se compara por medio del cociente del tiempo de computo proporcionado por ambas aproximaciones (Ecuación 6.1).

$$T = \frac{t_{BI1S}}{t_{RSR}} \tag{6.1}$$

En la Tabla 6.1 se observa que *RSR* reduce como media un 4.1% la longitud física de las redes respecto al *AEM*. *BI1S* reduce esta misma longitud en un 10.5%. Sin embargo *RSR* es (en media) 462 veces más rápido que *BI1S* para el conjunto de redes exploradas. Este gran aumento en la velocidad de rutado de las redes es el que va a permitir medir la congestión local en la fase de ubicación con el fin de evitarla.

Tabla 6.1 *Comparación en longitud y tiempo de cómputo de los algoritmos BI1S y RSR*

Red	Longitud red			Tiempo (s)		M (AEM)		T
	AEM	*BI1S*	*RSR*	*BI1S*	*RSR*	*BI1S*	*RSR*	t_{BI1S}/t_{RSR}
s100t60a	479	433	458	11.94	0.03	9.6	4.4	398
s100t60b	501	445	480	12.64	0.03	11.2	4.2	421
s100t60c	504	457	473	11.92	0.03	9.3	6.2	398
s100t60d	517	458	489	13.21	0.04	11.4	5.4	330
s100t60e	477	426	463	12.67	0.03	10.7	2.9	422
s100t60f	483	431	461	12.31	0.03	10.8	4.6	410
s200t60a	668	600	645	91.11	0.15	10.2	3.4	607
s200t60b	696	617	669	91.81	0.11	11.4	4.1	834
s200t60c	686	619	662	92.19	0.13	9.8	3.5	709
s200t60d	670	598	658	91.94	0.10	10.7	1.8	919
s200t60e	649	583	626	90.74	0.12	10.2	3.5	756
s200t60f	697	624	660	92.76	0.10	10.5	5.3	927

El algoritmo *RSR* se puede modificar ligeramente para mejorar la calidad de las soluciones proporcionadas. Así, se puede alterar el orden inicial en el que se disponen los nodos para su cableado con el objetivo de reducir la longitud del *ASMR*. En el algoritmo *RSR* original, se ordenan los nodos destino de acuerdo con su distancia al nodo fuente. Para reducir la longitud *ASMR*, podemos elegir como primer nodo, el más cercano al *centro geométrico* del conjunto de puntos que componen la red. Como siguiente nodo, se elige sucesivamente el nodo más cercano al último punto ordenado. Esta ordenación no modifica la complejidad del algoritmo ($O(n^2)$), y por tanto no varía significativamente el tiempo de cómputo. La Tabla 6.2 muestra la comparación entre los resultados proporcionados por este algoritmo *RSR* modificado *(RSRm),* y los producidos por la herramienta *BI1S*. Se puede observar que *RSRm* permite mejorar la longitud del

ASMR en un 6.6% respecto a la proporcionada por el *AEM*. En cuanto a la velocidad es 645 veces más rápido (en media) que *BI1S*.

Tabla 6.2 *Comparación en longitud y tiempo de cómputo de los algoritmos BI1S y RSRm*

Red	Longitud red			Tiempo (s)		M (AEM)		T
	AEM	*BI1S*	*RSRm*	*BI1S*	*RSRm*	*BI1S*	*RSRm*	t_{BI1S}/t_{RSRm}
s100t60a	479	433	450	11.94	0.02	9.6	6.1	597
s100t60b	501	445	456	12.64	0.04	11.2	9.0	316
s100t60c	504	457	470	11.92	0.02	9.3	6.7	596
s100t60d	517	458	479	13.21	0.03	11.4	7.4	440
s100t60e	477	426	457	12.67	0.02	10.7	4.2	634
s100t60f	483	431	452	12.31	0.03	10.8	6.4	410
s200t60a	668	600	617	91.11	0.11	10.2	7.6	828
s200t60b	696	617	640	91.81	0.12	11.4	8.0	765
s200t60c	686	619	658	92.19	0.10	9.8	4.1	922
s200t60d	670	598	617	91.94	0.12	10.7	8.0	766
s200t60e	649	583	612	90.74	0.12	10.2	5.7	756
s200t60f	697	624	659	92.76	0.13	10.5	5.5	714

Se pueden explorar otros ordenes, para los nodos destino, distintos a los considerados hasta ahora, obteniendo diferentes porcentajes de mejora respecto al *AEM*. Podemos conjeturar que la aplicación de la operación *Camino_Común* a una ordenación adecuada de los nodos, nos llevaría a la solución óptima. Sin embargo, es difícil saber si es posible encontrar esa ordenación *óptima* en tiempo polinómico. En la Figura 6.5, se presenta una tercera versión del algoritmo *RSR (RSRmm)* donde el orden en que se consideran los nodos no se establece al principio, sino antes de cada aplicación de la operación *Camino_Común*. En este caso, la complejidad del algoritmo crece a $O(n^3)$, mientras la calidad de las soluciones alcanza el 8.5% respecto al *AEM*, acercándose a la calidad 10.5% proporcionada por el algoritmo *BI1S*. La Tabla 6.3 muestra la

comparación de las soluciones y tiempos de computo para las redes de prueba entre los algoritmos *RSRmm* y *BIIS*. Aunque la complejidad del algoritmo RSRmm ha crecido a $O(n^3)$, su velocidad todavía mejora la de *BIIS* en 14.6 veces.

RSRmm(net N) return lista de regiones
$\Re$: lista de regiones *begin* *Sea v_s el vértice fuente de la red N* *Sea T={ v_{di} } , i=1..n el conjunto de vertices destino de la red N* *Sea OR la región de salida de la red N* *Insertar OR en $\Re$* *for each $v_{di} \in T$* *while $T \neq \phi$ loop* *Seleccionar $v_{di} \in T$ / Distancia(v_{di} , $\Re$) es mínima* *Camino común ($\Re$, v_{di})* *Quitar v_{di} de T* *Insertar IR_i de M_i en $\Re$* *end loop* *Elige camino ($\Re$)* *return $\Re$* *end RSRmm*

Figura 6.5 *Algoritmo RSRmm*

Tabla 6.3 *Comparación en longitud y tiempo de cómputo de los algoritmos BI1S y RSRmm*

Red	Longitud red			Tiempo (s)		M (AEM)		T
	AEM	*BI1S*	*RSRmm*	*BI1S*	*RSRmm*	*BI1S*	*RSRmm*	t_{BI1S}/t_{RSRm}
s100t60a	479	433	442	11.94	0.99	9.6	7.7	12.1
s100t60b	501	445	453	12.64	0.88	11.2	9.6	14.4
s100t60c	504	457	468	11.92	0.82	9.3	7.1	14.5
s100t60d	517	458	477	13.21	0.85	11.4	7.7	15.5
s100t60e	477	426	436	12.67	0.84	10.7	8.6	15.1
s100t60f	483	431	442	12.31	0.83	10.8	8.5	14.8
s200t60a	668	600	611	91.11	6.20	10.2	8.5	14.7
s200t60b	696	617	630	91.81	6.22	11.4	9.5	14.8
s200t60c	686	619	628	92.19	6.16	9.8	8.5	15.0
s200t60d	670	598	609	91.94	6.25	10.7	9.1	14.7
s200t60e	649	583	591	90.74	6.14	10.2	8.9	14.8
s200t60f	697	624	639	92.76	6.19	10.5	8.3	15.0

6.3 Optimización de la ubicación en etapas orientada a evitar la congestión local (U-RSR)

En la sección anterior, se presentó el algoritmo *RSR* que proporciona una aproximación rápida al *Árbol de Steiner Mínimo Rectilíneo* para el cableado de redes multiterminal. Aunque la velocidad de cableado se incrementa notablemente respecto a algoritmos de tipo laberinto y a otras aproximaciones al *ASMR*, su uso para medir la congestión local en la etapa de ubicación puede resultar demasiado costoso en tiempo de cómputo si no se diseña una estrategia adecuada. Las características de adaptabilidad de la *Optimización Combinatoria Termodinámica (OCT)*, proporcionan la clave para

hacer viable esta aproximación. En esta sección se presenta el método que se ha seguido para hacer posible la optimización de la ubicación basada en medidas de la congestión.

Como se resaltó en el capítulo 4, el algoritmo de *OCT* facilita enormemente la optimización en etapas, donde cada etapa es guiada por una función de coste distinta. Por una parte, el algoritmo se autosintoniza con las distintas funciones de coste. Por otra parte, debido a que el algoritmo puede ajustar automáticamente la temperatura a la solución de partida, se consigue preservar la calidad de las soluciones generadas entre las diferentes etapas de la optimización. De esta forma, podemos refinar la búsqueda de soluciones aplicando sucesivamente las distintas funciones de coste de acuerdo con el compromiso *calidad/tiempo* que proporcionan.

El *semiperímetro* es la métrica más rápida conocida para aproximar la longitud de una red multiterminal. Por lo tanto, será ésta la función de coste que nos permitirá eliminar soluciones inferiores y acercar la búsqueda hacia regiones del espacio de soluciones donde se encuentra el óptimo global. Si $N=\{N_1,N_2, ...,N_L\}$ es la lista de redes del circuito, la primera función de coste vendrá dada por

$$\Phi_1 = \sum_{i=1}^{L} semiperimetro(N_i) \qquad (6.2)$$

Si tenemos en cuenta que el cableado final estará basado en el algoritmo *RSR*, se puede mejorar la estimación de cable total en una segunda fase de ubicación. El número total de cables consumidos para cablear el circuito se puede reducir minimizando la longitud hallada mediante un cableado rápido con el algoritmo *RSR*. Sea $\Re_i = RSR(N_i)$, $i=1..L$, y sea $\Re$ la lista de regiones compuesta por todas las $\Re_i$, ϕ_2 vendrá dada por

$$\Phi_2 = \sum_{R_j \in \Re} s(R_j) \qquad (6.3)$$

Estas dos primeras etapas de optimización van dirigidas a la reducción de la congestión global. La eliminación de este tipo de congestión permite un enorme ahorro de tiempo para la tercera y última etapa, cuyo objetivo es la reducción de la congestión local. Si O_k es el número de cables ocupados del segmento de canal e_k después de las fases de ubicación y cableado, los valores O_k deben ser menores o iguales que la anchura de canal W para que el circuito se pueda cablear. Por lo tanto, para satisfacer esta restricción, el primer objetivo de las fases de ubicación y rutado debe ser minimizar los valores de O_k que excedan la anchura de canal W. Para medir los valores de O_k se puede realizar un cableado inicial, con el algoritmo RSR, de la solución proporcionada por la segunda etapa, y actualizar el cableado para las redes que se vean afectadas por el movimiento de bloques en la ubicación con la *OCT*. Los excesos de ocupación o *Congestión local* (φ_3), se pueden minimizar eficientemente con la función de coste dada por la Ecuación 6.4. En lo que sigue, a esta estrategia de ubicación en etapas le denominaremos *U-RSR,* dado que es una ubicación optimizada para el algoritmo de cableado *RSR*.

$$\Phi_3 = \sum_{\forall e_k \in E/O_k > W} (W - O_k)^2 \tag{6.4}$$

La *congestión local*, tal y como se ha definido, se puede considerar como una medida de la dificultad para cablear el circuito con el algoritmo RSR. En las Tablas 6.4 y 6.5 se muestra la *congestión local* medida sobre dos conjuntos de circuitos de prueba para distintas ubicaciones:

a) Para una ubicación aleatoria (Inicial)

b) Para una ubicación optimizada con la herramienta *VPR* basada en la métrica del *semiperímetro*

c) Para una ubicación optimizada en etapas *(U-RSR)*

Tal y como se ha definido, el valor de la *congestión local* depende de la anchura de canal escogida W. Para hacer que este valor sólo dependa del circuito, y por tanto sea fácilmente comparable con otras herramientas, hemos tomado W con un valor igual al máximo número de entradas de los bloques que componen el circuito. Este valor, se puede considerar como un límite inferior para W, debido a que el número de entradas de los CLBs suele ser menor o igual que la anchura de canal de la FPGA.

Tabla 6.4 *Comparación de la congestión local de los segmentos de canal producida por un conjunto de circuitos de prueba para distintas ubicaciones: inicial, optimizada con VPR y optimizada con U-RSR*

CIRC.	*Congestión local*			*Características circuito*		
	INICIAL	*VPR*	*U-RSR*	*#CLBs*	*#I/O*	*Tamaño FPGA*
9symml	1851	334	53	70	10	9x9
alu2	10572	1285	337	143	16	16x16
apex7	4296	197	28	77	86	11x11
example2	13384	511	22	120	151	19x19
k2fix	91900	7209	3687	358	91	19x19
term1	1662	127	25	54	44	8x8
vda	26085	3043	1122	208	56	15x15
e64	61529	3123	1136	274	130	17x17

Tabla 6.5 *Comparación de la congestión local de los segmentos de canal producida por un conjunto de circuitos de prueba para distintas ubicaciones: inicial, optimizada con VPR y optimizada con U-RSR*

CIRC.	*Congestión local*			*Características circuito*		
	INICIAL	*VPR*	*U-RSR*	*#CLBs*	*#I/O*	*Tamaño FPGA*
alu4	960976	35031	15853	1522	22	40x40
apex2	1865249	67249	34719	1878	41	44x44
apex4	833175	57447	30094	1262	28	36x36
bigkey	1636071	12199	3759	1707	426	54x54
diffeq	1215171	12873	5579	1497	103	39x39
dsip	944628	8677	2001	1370	426	54x54
ex5p	679825	53522	31281	1064	71	33x33
frisc	6938436	120122	79167	3556	136	60x60
misex3	964838	43270	20067	1397	28	38x38
s298	1262955	19510	6892	1931	10	44x44
seq	1643851	63448	31810	1750	76	42x42
tseng	596126	6273	2458	1047	174	33x33

En las Tablas 6.4 y 6.5 se observa que la *congestión local* se reduce drásticamente con una primera etapa de optimización basada en la métrica del *semiperímetro* como la realizada por VPR. Por otra parte, se observa una nueva reducción de la *congestión local* mediante la aplicación de dos nuevas fases de optimización (U-RSR). La obtención de estos valores ha sido posible debido a la conjunción de las dos nuevas técnicas presentadas anteriormente: por una parte, la *Optimización Combinatoria Termodinámica* ha permitido preservar la calidad de las soluciones entre etapas. Por otra parte, la rapidez del algoritmo *RSR* ha hecho viable la medida de la congestión en la etapa de ubicación.

Para mostrar de forma gráfica los beneficios de la optimización en etapas facilitada por la flexibilidad de la *Optimización Combinatoria Termodinámica*, se ha ubicado el

circuito *e64.net* siguiendo dos estrategias distintas: en la primera, a partir de una solución inicial generada de forma aleatoria, se ha reducido la *congestión local* mediante *OCT*. En la segunda, se ha iniciado la minimización de la *congestión local* a partir de la solución proporcionada por el algoritmo *VPR* (*punto A*). Puesto que *VPR* trabaja con una estimación de la longitud de las redes basada en la métrica rápida del semiperímetro (que en el peor de los casos es O(n)), alcanzar el punto A es poco costoso computacionalmente comparado con el cálculo de la *congestión local* que implica un cableado *RSR* ($O(n^2)$). En la gráfica se pone de manifiesto el ahorro de tiempo obtenido en la reducción de la *congestión local (punto B*) al partir de una solución pre-optimizada (*punto A*), respecto a la optimización realizada (*punto C*) a partir de una ubicación aleatoria. Se debe destacar que la ganancia obtenida en la primera etapa, no se echa a perder debido a las propiedades adaptativas de la *Optimización Combinatoria Termodinámica.*

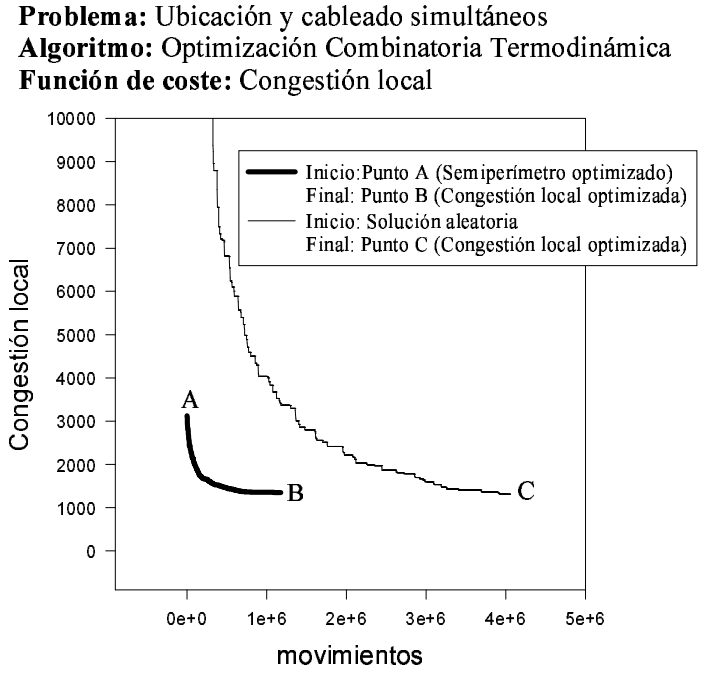


Figura 6.6 *La gráfica muestra la flexibilidad proporcionada por la Optimización Combinatoria Termodinámica para la optimización en etapas. Las soluciones C y B fueron obtenidas en una y dos etapas de optimización respectivamente.*

6.4 Cableado global basado en el algoritmo *RSR (CG-RSR)*

Terminada la fase de ubicación, la siguiente etapa del ciclo diseño físico es el cableado global. La estrategia de optimización en etapas proporciona algunas ventajas de cara a realizar el cableado global. Si tras la última fase de optimización de la ubicación, el valor de la *congestión local* es igual a cero, la congestión habrá sido eliminada. En este caso, el cableado provisional realizado con el algoritmo *RSR* en la fase de ubicación, será válido como cableado global. En caso contrario, el algoritmo *RSR* se debe combinar con alguna otra estrategia de forma que se negocien caminos alternativos para las redes afectadas por la congestión. Además, se debe proporcionar prioridades de rutado para las redes que componen el camino crítico de forma que no sufran desviaciones respecto al camino más directo. En esta sección se presenta la herramienta *CG-RSR* basada en el algoritmo *RSR* para realizar el cableado global del circuito sobre la FPGA partiendo de una ubicación dada.

Dado que el algoritmo *RSR* proporciona una buena aproximación al *Árbol de Steiner Mínimo Rectilíneo* para cada red, el primer paso que realiza la herramienta *CG-RSR* consiste en calcular las regiones *RSR* para todas las redes. Aunque la mayoría de las redes podrán ser rutadas a través de esas regiones, pueden existir zonas de congestión que impidan el cableado para algunas de ellas, de forma que tengan que ser desviadas. Por lo tanto, es importante establecer un orden de prioridades para el rutado de las redes de forma que aquellas que componen el camino crítico no sean desviadas. Por otra parte, podemos también dar prioridad a las redes cuya dificultad para ser rutadas sea grande. Sea $N=\{N_1, N_2,..., N_L\}$ la lista de redes del circuito. Si *R* es una región *RSR* determinada para rutar una red, y $|R|$ es el número de arcos e_k tal que $e_k \in R$, la *dificultad* (δ) para rutar *R* puede estimarse por medio de la Ecuación 6.5, donde O_k es la ocupación del segmento de canal correspondiente al arco e_k. Con esta definición, un valor alto de δ indica que *R* transcurre por zonas muy congestionadas.

$$\delta = \frac{\sum_{e_k \in R} O_k^2}{|R|} \tag{6.5}$$

Por lo tanto, en un primer paso, las regiones *RSR* se ordenan de acuerdo con su *prioridad* y *dificultad* de rutado estimada. La herramienta *CG-RSR* trata de rutar las regiones *RSR* (lista $\Re$) tomadas en orden. En caso de éxito en el cableado de una región correspondiente a la red N_i, la región se mueve de la lista $\Re$ de regiones sin rutar a la lista de regiones rutadas $\mathcal{R}_i$ correspondiente a la red N_i. Para las regiones cuyo camino se encuentra saturado, se aplica una estrategia de tipo laberinto *("maze router")* limitando el número de desviaciones. Cuando no hay éxito, y el tamaño de la región más el número de desviaciones es mayor o igual que la distancia desde el nodo destino a otros puntos ya rutados pertenecientes a la misma red, se aplica una estrategia de tipo laberinto desde esos puntos. El número de desviaciones se va incrementando gradualmente hasta alcanzar el máximo número de desviaciones permitido. El algoritmo termina con éxito cuando se vacía la lista de regiones $\mathcal{R}$ que quedan por rutar. Si se alcanza el máximo número de desviaciones permitidas sin vaciar la lista, no se habrá encontrado una solución para el cableado global del circuito con la anchura de canal correspondiente a la FPGA objetivo. En este caso, tendremos que recurrir a una FPGA de mayor tamaño o que disponga de más recursos de cableado.

```
CG-RSR (netlist N,recursos cableado G(V,E)) return rutado de N
    ℜ:lista de regiones: =∅                    -- mantiene las regiones que quedan por rutar
    ℜi : lista de regiones:=∅   i=1..L          -- mantiene las regiones relacionadas con la red i
    camino_con_éxito:list of regions:= ∅        -- región rutada con éxito
begin
    for i=1 to L
    loop
        ℜi = RSR (Ni)
        Insertar ℜi en ℜ
        ℜi = ∅     --se vacia para guardar sólo las regiones rutadas con éxito
    end loop
    Ordenar ℜ de acuerdo con las prioridades y dificultad estimada δ
    for each Rj ∈ ℜ          -- RSR routing
    loop
        camino_con_éxito = intentar_rutado(Rj )     --inserta Rj en camino_con_éxito si hay éxito
        if camino_con_éxito ≠ ∅
        then
            Ok= Ok +1  ∀ ek ∈ Rj
            Quitar Rj de ℜ
            Mover camino_con_éxito a ℜi    -- región de la red i rutada con éxito
        end if
    end loop
--Para las regiones sin éxito en el rutado directo se aplica una estrategia  "Maze routing"
    Desviaciones=0
    while (ℜ ≠ ∅ and Desviaciones<MaxDesviaciones)
        loop
            for each Rj of ℜ
            loop
                deviaciones=deviaciones+1
                 camino_con_éxito=maze_ router(Rj ,desviaciones) --inserta en camino_con_éxito el camino
                 alternativo
                if camino_con_éxito=∅              -- si no hay éxito
                then
                    vdj=destino(Rj)
                    dj=distancia(vdj , ℜi)
                    if (dj ≤ s(Rj )+ deviaciones)
                    then
                        R'= R( Vértice_más_cercano (vdj , ℜi ) ,vdj)
                        camino_con_éxito = maze_ router(R' ,s(Rj )+deviaciones-dj )
                    end if
                end if
                if camino_con_éxito  ≠ ∅      --si hay éxito
                then
                        Ok=Ok +1  ∀ ek ∈ camino_con_éxito
                        quitar Rj from ℜ
                        mover camino_con_éxito a ℜi
                end if
            end loop
        end loop
    return  ℜi  , i=1 .. L
end CG-RSR
```

Figura 6.7 *Herramienta de cableado global CG-RSR*

6.5 Comparación de resultados entre *VPR* y *UCG-RSR*

Como hemos visto en la sección anterior, la herramienta de cableado global *CG-RSR* está basada en el algoritmo *RSR*. Por lo tanto, los mejores resultados los podrá proporcionar en el cableado de ubicaciones proporcionadas por la herramienta *U-RSR* (sección 6.3). A esta estrategia conjunta de ubicación y cableado global le llamaremos *UCG-RSR*. En esta sección se realiza una comparación de los resultados producidos por las herramientas *VPR* y *UCG-RSR* en la ubicación y rutado global sobre un conjunto de circuitos de prueba.

En el diseño físico orientado a FPGAs, es habitual medir la calidad de las soluciones que proporciona una herramienta, por medio del valor de la anchura de canal necesaria *(WN)* para ubicar y cablear un circuito. Las Tablas 6.6 y 6.7 muestran el valor de *WN* para una ubicación aleatoria inicial de un conjunto de circuitos de prueba. Asimismo, se compara el valor de *WN* obtenido por la herramienta *VPR*, con el obtenido por UCG-RSR. Como se comentó en el capítulo 2, la herramienta *VPR* realiza una ubicación con el algoritmo de *Enfriamiento Simulado*, guiado por la función de coste *Bounding-Box*. Para el cableado utiliza una variación del algoritmo *Pathfinder negotiated congestion* [EMHB95], el cual está basado en técnicas de "*laberinto*" y "*ripping-up and re-routing*". Los valores de *WN* proporcionados por la herramienta *VPR* constituyen un excelente punto de referencia, ya que *VPR* ha mejorado apreciablemente [BR97] la anchura de canal demanda por otras herramientas de ubicación y cableado como GCE, SEGA , GBP, OGC, IKMB, TRACER, FPR.

En las Tablas 6.6 y 6.7 se observa que la estrategia *UCG-RSR* proporciona valores de *WN* menores o iguales que *VPR* para todos los circuitos de prueba. El tiempo invertido por *U-RSR* en la ubicación es mayor que el invertido por *VPR* puesto que *U-RSR* realiza optimizaciones adicionales para evitar la congestión local. Sin embargo, la herramienta de cableado *CG-RSR* invierte menos tiempo de *CPU* que *VPR* debido a que parte del cableado se lleva a cabo a través de las regiones determinadas por *U-RSR* durante la etapa ubicación.

Tabla 6.6 *Comparación del ancho de canal demandado (WN) proporcionado por UCG-RSR y VPR para un conjunto de circuitos de prueba*

CIRC.	*Anchura de canal demanda (WN)*			*Características circuito*		
	INICIAL	*VPR*	*UCG-RSR*	*#CLBs*	*#I/O*	*Tamaño FPGA*
9symml	13	6	5	70	10	9x9
alu2	18	6	6	143	16	12x12
apex7	16	6	5	77	86	11x11
example2	27	6	4	120	151	19x19
k2fix	32	8	8	358	91	19x19
term1	16	5	5	54	44	8x8
vda	22	7	6	208	56	15x15
e64	28	7	6	274	130	17x17

Tabla 6.7 *Comparación del ancho de canal demandado (WN) proporcionado por UCG-RSR y VPR para un conjunto de circuitos de prueba*

CIRC.	*Anchura de canal demanda (WN)*			*Características FPGA*		
	INICIAL	*VPR*	*UCG-RSR*	*#CLBs*	*#I/O*	*Tamaño*
alu4	52	8	8	1522	22	40x40
apex2	49	9	9	1878	41	44x44
apex4	48	10	10	1262	28	36x36
bigkey	46	6	6	1707	426	54x54
diffeq	45	7	7	1497	103	39x39
dsip	41	7	6	1370	426	54x54
ex5p	43	11	10	1064	71	33x33
frisc	69	11	10	3556	136	60x60
misex3	46	9	8	1397	28	38x38
s298	49	7	6	1931	10	44x44
seq	55	9	8	1750	76	42x42
tseng	43	7	6	1047	174	33x33

6.6 Conclusiones

El problema de la congestión local se ha afrontado tradicionalmente únicamente en la etapa de cableado. El elevado coste computacional, incluso para pequeños circuitos, de las tentativas para hacer una ubicación y cableado simultáneos las han convertido en poco atractivas.

En este capítulo, se ha presentado el algoritmo *RSR* que proporciona una aproximación rápida al *ASMR*. La velocidad de este algoritmo, junto con el método *OCT*, ha permitido diseñar una estrategia de optimización en etapas (*U-RSR)* para el refinamiento de las soluciones evitando la congestión. Asimismo, se ha desarrollado una herramienta de cableado global basada en el algoritmo *RSR* (*CG-RSR*) que se beneficia de la buena distribución, proporcionada por *U-RSR,* de los bloques lógicos que componen las redes.

La detección y eliminación temprana de la *congestión local* con la estrategia *U-RSR,* se ha puesto de manifiesto para un conjunto de circuitos de prueba. La anchura demanda por estos circuitos se ha reducido en un 60% de los casos, respecto a la proporcionada por una herramienta de calidad contrastada como es *VPR*.

Capítulo 7

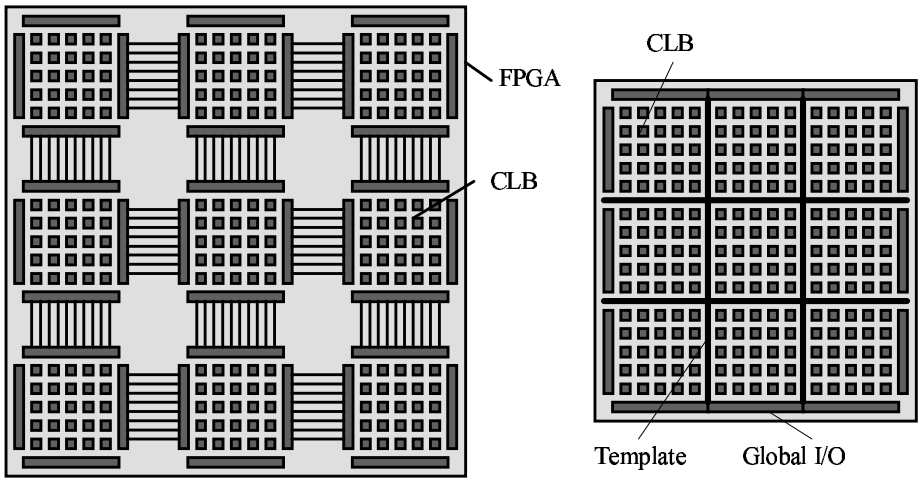
Método *PUR* para la partición de circuitos en Sistemas Multi-FPGA con topología de malla

7.1 Introducción

Cuando un circuito excede la capacidad de una única FPGA, el diseñador debe recurrir a un sistema Multi-FPGA. Como se describió en el capítulo 2, la escasez de pines de entrada/salida de las FPGAs provoca una infrautilización de los recursos lógicos que ofrecen los sistemas Multi-FPGA. Esta escasez limita enormemente la capacidad de comunicación entre las FPGAs que constituyen el sistema. Por lo tanto, una partición del circuito en bloques que tenga en cuenta las fuertes carencias de comunicación entre FPGAs, debe ser el objetivo prioritario en el ciclo de diseño físico de circuitos orientado a sistemas Multi-FPGA. La etapa de partición ha sido objeto de múltiples estudios considerando distintos objetivos. Así, por ejemplo, en el método propuesto por Kuznar [KBK93] para sistemas heterogéneos, el coste económico final es el principal objetivo. En [HPLT98] y [CSZ96] la rutabilidad interna de las FPGAs es considerada mediante distintos modelos. Fang [FW98] realiza un particionamiento teniendo en cuenta información de los retardos y de la estructura de diseño.

En la mayoría de estos estudios, se asume que no existen restricciones sobre qué particiones se pueden comunicar, de forma que sólo serán adecuados para topologías que utilizan FPICs, donde la capacidad de comunicación entre todas las FPGAs esté equilibrada. En este capítulo, nos centraremos en los sistemas Multi-FPGA con topología de malla, donde este tipo de partición puede complicar, e incluso imposibilitar, las etapas posteriores del diseño físico. Durante el resto del capítulo haremos referencia a estos sistemas como Mallas de FPGAs (MFPGAs). Las MFPGAs despiertan un gran interés debido a la simplicidad de su red de interconexión (Figura 7.1(a)). En estos sistemas, las FPGAs se conectan formando una malla, es decir, las FPGAs están conectadas únicamente a las FPGAs vecinas más cercanas. Por lo tanto,

las herramientas de partición deben contemplar las restricciones de comunicación entre las FPGAs que se encuentran alejadas unas de otras. Por ejemplo, deben tener en cuenta el consumo adicional de pines de E/S producido en las FPGAs intermedias al realizar ese tipo de conexiones. Por otra parte, las señales de E/S globales del circuito no pueden asignarse a FPGAs internas puesto que los pines de E/S de estas FPGAs están completamente ocupados constituyendo la red. Por lo tanto, las señales globales de E/S se deben asignar cuidadosamente entre las FPGAs que puedan soportarlas.



(a) Multi-FPGA con topología de malla *(b) Modelo SFPGA*

Figura 7.1 *Sistema Multi-FPGA y su modelo simplificado*

La aproximación más natural para cumplir con todas estas restricciones puede ser la ubicación y el rutado simultáneos. En [Hau94], Hauck reconoce que un planteamiento para proporcionar soluciones de calidad en la implementación de circuitos sobre sistemas Multi-FPGA, podría ser una *partición y ubicación global* simultáneas guiadas por un *Enfriamiento Simulado*, como el desarrollado en [RS93,Vij90]. De esta forma, se conoce qué elementos lógicos están asignados a qué FPGAs, y por tanto, se pueden determinar las zonas congestionadas mediante un cableado de las señales correspondiente a bloques lógicos ubicados en distintas FPGAs. Sin embargo, Hauck considera que este proceso

puede llegar a ser muy lento al realizarse el cableado para evaluar el coste de cada configuración. Esta observación es correcta cuando, por una parte, se utilizan los habituales algoritmos de tipo *laberinto* para realizar el cableado, y por otra, cuando se realiza el cableado durante todo el proceso de optimización de la ubicación. Sin embargo, la aproximación de una *partición y ubicación global* simultáneas se puede hacer viable mediante una optimización en dos etapas, donde sólo en la última etapa se recurre a un algoritmo de cableado rápido como es *RSR* (Capítulo 6), para aliviar las zonas congestionadas.

En este capítulo, se describe un método de *partición* apoyada en la *ubicación y rutado simultáneos (PUR)* para el diseño físico sobre sistemas MFPGA. *PUR* realiza la partición del circuito después de equilibrar la demanda de comunicación real entre las FPGAs mediante una ubicación y rutado simultáneos posibilitado por las técnicas de *Optimización Combinatoria Termodinámica* y *Regiones de Steiner Rectilíneas* descritas en los capítulos precedentes. Con este método, los bloques lógicos se distribuyen sobre las FPGAs de tal forma que se armoniza el número de redes que cruzan las fronteras entre FPGAs, con los recursos disponibles. El método *PUR* se puede aplicar para resolver dos tipos de problemas, la implementación sobre un sistema MFPGAs predeterminado, o el diseño del sistema MFPGAs que mejor se ajusta a un determinado circuito. En este capítulo se ha aplicado el método *PUR* para determinar la MFPGAs que mejor se ajusta a un conjunto de circuitos de prueba, obteniéndose interesantes porcentajes de utilización de los recursos lógicos para algunos de estos circuitos.

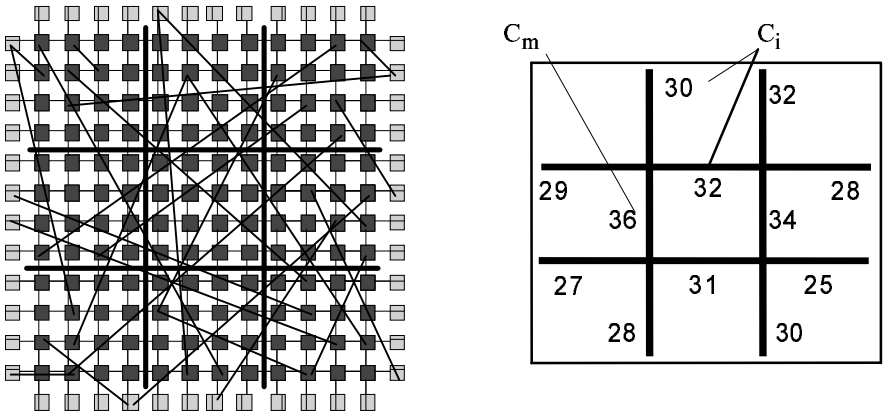
En la siguiente sección se presenta un modelo para describir los sistemas MFPGAs, así como la formulación del problema de partición y ubicación dentro de ese modelo. En la sección 7.3 se describe la estrategia diseñada para obtener un buen compromiso *tiempo/calidad* para la resolución del problema con el algoritmo de *Optimización Combinatoria Termodinámica*. En la sección 7.4 se presentan los resultados experimentales obtenidos en la partición de un conjunto de circuitos de prueba. El capítulo termina con las conclusiones en la sección 7.5.

7.2 Modelo para los sistemas MFPGAs y formulación del problema

En un sistema MFPGAs, las FPGAs están conectadas a sus vecinas a través de sus pines de E/S (Figura 7.1(a)). Para nuestro propósito, los sistemas MFPGAs se pueden modelar como una gran única FPGA (SFPGA), donde las fronteras entre FPGAs vecinas queden difuminadas y se representen mediante una plantilla T superpuesta constituida por un conjunto de segmentos $T=\{T_1,..,T_n\}$ (Figura 7.1(b)). Sea C_i el número de redes que atraviesan la frontera T_i después de la ubicación y del cableado. Se define el *conjunto de corte* a través de T como el conjunto $C=\{C_i\}$ para i=1.. n, y se define C_m como el mayor de los elementos de C. El objetivo de las herramientas de ubicación y rutado debe ser la optimización del *conjunto de corte* C para satisfacer las restricciones de comunicación entre las FPGAs.

Podemos distinguir básicamente dos planteamientos para implementar circuitos en sistemas MFPGA. En el primero, *Malla Fija (MF),* el número y tamaño de las FPGAs así como su disposición está determinada por la arquitectura del sistema. En este caso, el modelo del sistema MFPGAs se construye juntando las FPGAs componentes en una gran SFPGA y guardando las fronteras de separación en una plantilla (Figura 7.1(b)). Después de la optimización de la ubicación y del rutado, el circuito se ajustará a este sistema MFPGA si el *conjunto de corte* (valores C_i) satisface las restricciones de comunicación entre FPGAs (Figura 7.2(b)). En el segundo planteamiento, se ha de determinar el sistema MFPGA que mejor se ajusta para un circuito dado. Por lo tanto, se explora el número de FPGAs, su disposición y tamaño necesarios para albergar el circuito en el sistema MFPGA. A lo largo del resto del capítulo nos referiremos a esta aproximación como *Malla Adaptada (MA).* En esta aproximación, el modelo del sistema MFPGA se puede construir con una SFPGA de tamaño mínimo suficiente para albergar el circuito, y superponiendo una plantilla que modele la partición del circuito (Figura 7.2(a)). Después de optimizar el *conjunto de corte* en la etapa de ubicación, el valor de C_m fijará el tamaño de las FPGAs necesarias (NFPGA) y la plantilla o esquema de partición su disposición (Figura 7.2(b)). Suponiendo que el número de pines de E/S para una FPGA de NxN CLBs es 2N, el tamaño de las NFPGAs tendrá que ser al menos $(C_m/2)$ x $(C_m/2)$ para valores pares de C_m, ó $(C_m+1)/2$ x $(C_m+1)/2$ para valores impares

de C_m. La partición será válida si las NFPGAs resultantes no son mayores que las FPGAs disponibles. Cuando esta restricción no se satisface, son necesarias FPGAs adicionales, y por lo tanto, se debe explorar un esquema de particionamiento de *grano más fino*. Debido a la escasa capacidad de comunicación entre FPGAs, la comunicación externa entre FPGAs es mucho más costosa que la comunicación interna. Por lo tanto, particiones grandes que cumplan las restricciones de tamaño impuestas por la disponibilidad de componentes en las librerías, conducen a mayores porcentajes de utilización de los CLBs de las FPGAs.



(a) Ubicación *(b) Conjunto de corte (C) después del cableado*

***Figura** 7.2 Ubicación sobre el modelo SFPGA*

7.3 Optimización del *conjunto de corte* mediante el método *PUR*

En esta sección se describe el método de partición *PUR*, el cual optimiza el conjunto de corte para adecuarlo a la capacidad de comunicación real entre FPGAs. El método *PUR* esta basado en una ubicación y cableado simultáneos. Mediante esta técnica, es posible conocer el conjunto de corte en todo momento, y por tanto, puede ser optimizado. Considerando el modelo descrito en la sección previa, se pueden aplicar las técnicas comunes para la ubicación y el rutado de circuitos sobre la SFPGA con el objetivo de satisfacer las duras restricciones de comunicación. Como se describió en los capítulos precedentes, la *Optimización Combinatoria Termodinámica* es un método de optimización combinatoria suficientemente flexible para adaptarse a nuevas funciones de coste y estrategias de optimización. Nuestro método realiza dos fases de *Optimización Combinatoria Termodinámica* para la ubicación de los bloques lógicos con el fin de reducir el *conjunto de corte*.

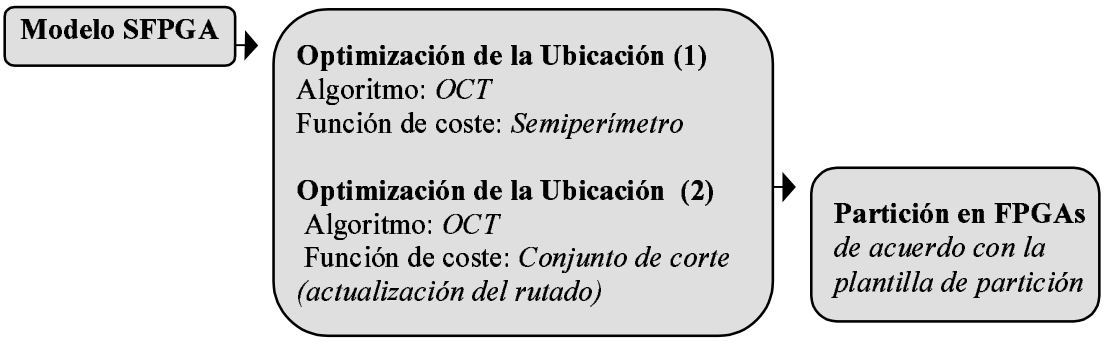


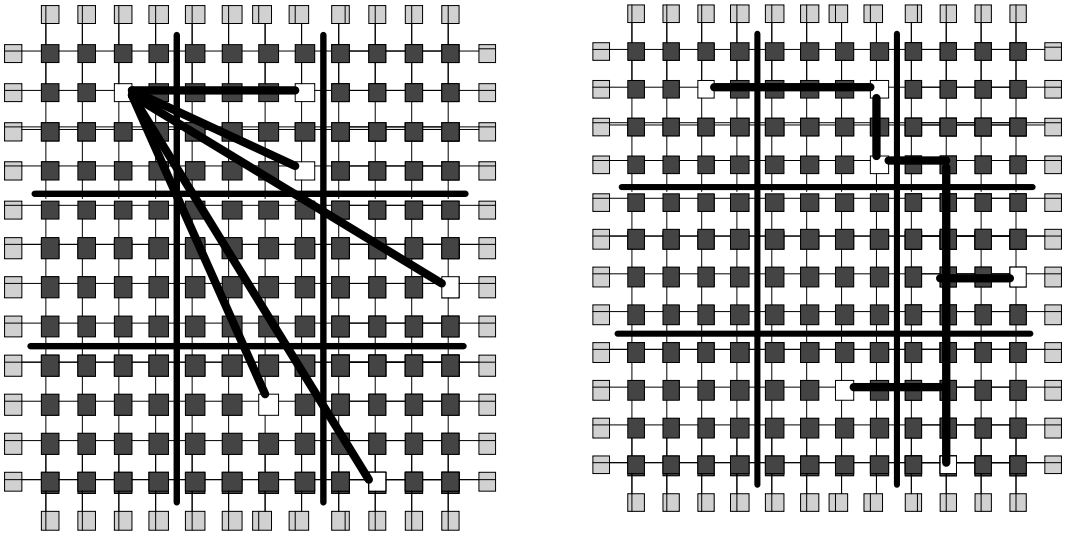
Figura 7.3 *Visión general del método PUR*

Funciones de coste para la ubicación en etapas mediante *OCT*

La capacidad de comunicación entre FPGAs es la restricción más dura que se presenta en los sistemas Multi-FPGA. El objetivo de nuestra herramienta de ubicación será distribuir las redes del circuito en el modelo SFPGA, de forma que el *conjunto de corte* producido en la plantilla se ajuste a la capacidad de comunicación entre FPGAs vecinas. Para lograr esta optimización del *conjunto de corte* la ubicación con *OCT* se realiza en dos etapas. La primera etapa está guiada por la función de coste rápida del *semiperímetro* (ϕ_1) (Ecuación 7.1), la cual aproxima la longitud de una red por la mitad del perímetro (P_i) del rectángulo que la engloba. Con esta función de coste, los bloques lógicos que están relacionados se agrupan en la misma región de la SFPGA. Así, la longitud de cable total y por tanto el *conjunto de corte* a través de la plantilla de partición se reducen.

$$\phi_1 = \frac{1}{2}\sum_{i \in nets} P_i \qquad (7.1)$$

Aunque en esta fase se consigue una alta reducción en el *conjunto de corte* (ver Tabla 7.3), es posible realizar mejoras adicionales sobre estos valores críticos. Para ser capaz de minimizar el número de redes que cruzan la plantilla de partición es necesario estimar o medir ese número. La precisión que podemos obtener estimando el *conjunto de corte* en la etapa de ubicación es limitada, debido a que el *conjunto de corte* depende de forma importante del proceso de cableado. La Figura 7.4a muestra un ejemplo de la ubicación de los elementos que componen una red multiterminal, y de los posibles cruces que se producirán sobre la plantilla de partición cuando se realice el cableado. Se puede observar que la gran indeterminación sobre los caminos finales hace difícil la estimación del *conjunto de corte*. La Figura 7.4b muestra el mismo ejemplo una vez que se ha realizado el cableado de la red multiterminal. En este caso, la indeterminación se ha resuelto en favor de una alternativa de forma que el *conjunto de corte* se puede medir, y a partir de ahí, minimizar.



(a) Indeterminación del cableado (posible conjunto de corte para una red multiterminal)

(b) Determinación del cableado (después del cableado con RSR, el conjunto de corte puede ser medido y por tanto minimizado)

Figura 7.4 *Conjunto de corte para una red multiterminal*

La segunda etapa de la optimización de la ubicación tiene en cuenta estas consideraciones, de forma que realiza un cableado inicial, y actualiza las redes afectadas debido al movimiento de bloques en cada iteración de la fase de ubicación con *OCT*. La actualización del cableado de las redes afectadas podría ser muy costoso en tiempo si se utilizan las estrategias habituales de tipo *laberinto* para el cableado. Sin embargo, el desarrollo de aproximaciones rápidas del *Árbol de Steiner Mínimo Rectilíneo* (*ASMR*) como las proporcionadas por el algoritmo *RSR*, permiten la integración del cableado en el proceso de optimización de la ubicación. Así, nosotros utilizamos el algoritmo de cableado *RSR* descrito y usado en capítulos precedentes, para actualizar el cableado en cada iteración del proceso de optimización de la ubicación. Por otra parte, para el estudio de la demanda de comunicación entre FPGAs, se pueden ignorar los detalles correspondientes a los recursos de interconexión internos de las FPGAs. Por lo tanto,

con el objetivo de acelerar el proceso de cableado, los recursos de interconexión internos de las FPGAs se modelan mediante una simple malla rectilínea de CLBs (Figura 7.4). Como hemos visto en capítulos precedentes, el algoritmo de cableado *RSR* obtiene aproximaciones de calidad al *ASMR*, es decir, busca caminos de longitud mínima entre el nodo fuente y los nodos destino. Con ello se consigue un doble objetivo: reducir la congestión media y los retardos de las redes.

Sea G(V, E) el grafo que representa una malla rectilínea, donde $V=\{v_1,..., v_{|V|}\}$ es el conjunto de CLBs del modelo SFPGA, y sea $E=\{e_1,.., e_{|E|}\}$ el conjunto de arcos que conectan los CLBs vecinos. Sea T la plantilla de partición superpuesta sobre G (Figura 7.2(a)). El *conjunto de corte C* (Figura 7.2(b)) se puede medir en cada iteración de la ubicación después de actualizar el rutado sobre G (Figura 7.4b). La función de coste (ϕ_2) en esta segunda etapa de optimización depende del problema tratado, *Malla Fija* o *Malla Adaptada*. En la aproximación de *MF* el objetivo principal es satisfacer las restricciones de comunicación. Así, si *S* es el número de cables disponibles para comunicar dos FPGAs vecinas, los valores del *conjunto de corte* por encima del nivel S se deben minimizar. La minimización se puede realizar eficientemente por medio del primer término de la Ecuación 7.2, que penaliza de forma cuadrática los excesos de demanda de comunicación entre FPGAs (valores Ci) sobre la capacidad disponible para cada comunicación (S). En *MA*, el valor de C_m determina el tamaño de las FPGAs necesarias. Para esta aproximación, el valor de C_m se minimiza por medio del primer término de la Ecuación 7.3. El segundo término de la función de coste ϕ_2, tanto para *MF* como para *MA* (Ecuaciones 7.2, 7.3), se encarga de minimizar las ocupaciones O_j de los arcos $e_j \in E$, que excedan la anchura de canal (W) de las FPGAs. Puesto que el rutado ya se actualiza en cada iteración para obtener el primer término de ϕ_2, este cálculo adicional para el segundo término, no supone un esfuerzo computacional significativo. Este segundo término de la función de coste ϕ_2, es una medida de la congestión local, y permite distribuir uniformemente las redes en el modelo SFPGA para mejorar la rutabilidad interna de las FPGAs.

$$\Phi_2 = \sum_{C_i \in C \,/\, C_i > S} (C_i - S)^2 + \sum_{e_j \in E \,/\, O_j > W} (O_j - W)^2 \qquad \textit{(para MF)} \qquad (7.2)$$

$$\Phi_2 = \sum_{C_i \in C} C_i^2 + \sum_{e_j \in E \,/\, O_j > W} (O_j - W)^2 \qquad \textit{(para MA)} \qquad (7.3)$$

7.4 Aplicación del método *PUR* a circuitos concretos

Existe un gran número de estudios publicados relacionados con la partición de circuitos orientada a sistemas Multi-FPGA, pero en pocos de ellos se realizan las fases de ubicación y rutado globales para poner a prueba sus resultados. En otras aproximaciones [OG95], se realiza el particionamiento para determinadas arquitecturas, haciendo difícil las comparaciones directas.

Nosotros hemos puesto a prueba el método de optimización *PUR*, descrito en este capítulo, para un sistema MFPGA con un conjunto de circuitos de prueba [BR98]. La Tabla 7.2 muestra las características de estos circuitos, así como el tamaño del modelo SFPGA necesario para albergar cada circuito. Para destacar el grado de rutabilidad de estos circuitos, también se muestra la anchura de canal demandada *(WN)*, suponiendo que la implementación del circuito se realizase sobre una gran hipotética FPGA (utilizando sus recursos generales de interconexión). Para calcular este valor se ha utilizando la herramienta de ubicación y rutado *VPR*. Asimismo, se asume la disponibilidad de las FPGAs de Xilinx descritas en la Tabla 7.1.

Tabla 7.1 *Tamaños de las FPGAs elegidas para el estudio*

DISPOSITIVOS	*CLB Matriz*	*CLBs*	*IOBs*
XC4013XLA	24x24	576	192
XC4020XLA	28x28	784	224
XC4028XLA	32x32	1024	256

Para proporcionar un criterio de comparación directa con futuros estudios, nuestra herramienta calcula el porcentaje de utilización de CLBs máximo conseguido para los circuitos de prueba. Para obtener este coeficiente, se aplica la aproximación de *Malla Adaptada* descrita en la sección 7.2. Así, se puso a prueba inicialmente un esquema de particionamiento de 2x2 para todos los circuitos. La Tabla 7.3 describe la evolución del valor de C_m durante la optimización de la ubicación para esa plantilla de partición. La segunda columna muestra el *máximo del conjunto de corte* para una ubicación inicial aleatoria, y un cableado realizado con el algoritmo *RSR*. En la tercera columna puede observarse el valor de C_m después de la fase de optimización con el *semiperímetro* (ϕ_1). La fase posterior de optimización y rutado simultáneos (ϕ_2) conduce al valor final de C_m mostrado en la cuarta columna.

Tabla 7.2 *Circuitos de prueba: #CLBs, #E/S, tamaño y anchura de canal demandados suponiendo que los circuitos fuesen implementados en una única gran FPGA*

CIRC	*#CLBs*	*#E/S*	*Tamaño SFPGA*	*WN*
tseng	1047	174	33x33	7
ex5p	1064	71	33x33	11
apex4	1262	28	36x26	10
misex3	1397	28	38x28	9
diffeq	1497	103	39x29	7
alu4	1522	22	40x40	8
seq	1750	76	42x42	9
s298	1931	10	44x44	6
apex2	1878	41	44x44	9
dsip	1370	426	54x54	7
bigkey	1707	426	54x54	7

Tabla 7.3 *Evolución del máximo de conjunto de corte (C_m) (partición de 2x2)*

CIRC.	C_m *(i)*	C_m (ϕ_1)	C_m (ϕ_2)
tseng	435	67	45
ex5p	439	153	125
apex4	473	168	135
misex3	533	142	105
diffeq	619	97	64
alu4	520	129	94
seq	679	151	113
s298	621	98	53
apex2	774	163	121
dsip	575	31	20
bigkey	730	40	27

Analizando los resultados de la Tabla 7.3 podemos observar que, con la fase ϕ_1, se consigue una gran reducción en el *conjunto de corte* para todos los circuitos. La función de coste del *semiperímetro* permite agrupar los bloques lógicos relacionados en una misma área. La segunda fase de la optimización, aunque es más lenta debido a la actualización del cableado en cada iteración de la ubicación, permite una nueva reducción en el crítico valor de C_m.

El valor de C_m determina el tamaño mínimo de las NFPGAs (sección 7.2). Puesto que hemos supuesto la disponibilidad de las FPGAs descritas en la Tabla 7.1, el número máximo de pines de E/S que comunican dos FPGA vecinas es 64 (2x32). Por lo tanto, sólo los circuitos con un valor de C_m menor o igual que 64 aceptan un esquema de partición de 2x2. La Tabla 7.4 muestra el tamaño de las FPGAs necesarias (NFPGAS) y el porcentaje de utilización de CLBs *(PU)* (Ecuación 7.4) para estos circuitos.

$$PU = 100x\frac{CLBs\ utilizados}{CLBs\ totales} \tag{7.4}$$

donde CLBs totales=Particiones x tamaño(NFPGA) y CLBs utilizados=CLBs del circuito

Tabla 7.4 *Esquema de partición de 2x2. Para todos los circuitos que aceptan este esquema de partición se muestra: el valor máximo de conjunto de corte (C_m), el tamaño de las FPGAs necesarias para soportar este conjunto de corte (NFPGAs), y el porcentaje de utilización (PU)*

CIRC.	C_m (ϕ_2)	#Particiones	Disposición NFPGA array	Tamaño NFPGAs	PU
s298	53	4	2x2	28x28	62 %
bigkey	27	4	2x2	28x28	55 %
tseng	45	4	2x2	24x24	45 %
dsip	20	4	2x2	28x28	44 %
diffeq	64	4	2x2	32x32	37 %

Cuando el valor de C_m es mayor de 64 se debe probar un esquema de partición de grano más fino, repitiendo la segunda fase de ubicación con *OCT*. La Tabla 7.5 muestra los circuitos que aceptan esquemas de particionamiento de 3x3 y 4x4.

Tabla 7.5 *Esquemas de partición de 3x3 y 4x4. Para todos los circuitos que aceptan estos esquemas de partición se muestra: el valor máximo de conjunto de corte (C_m), el tamaño de las FPGAs necesarias para soportar este conjunto de corte(NFPGAs), y el porcentaje de utilización (PU).*

CIRC.	***C_m (ϕ_2)***	***#Particiones***	***Disposición array NFPGAs***	***Tamaño NFPGA***	***PU***
alu4	60	9	3x3	32x32	17%
misex3	59	9	3x3	32x32	15%
seq	56	16	4x4	28x28	14%
apex2	58	16	4x4	32x32	11%
apex4	60	16	4x4	32x32	8 %
ex5p	60	16	4x4	32x32	6 %

El *PU* final obtenido para todos los circuitos se extiende desde el 6% al 62%, con un porcentaje promedio de 28. En la Tabla 7.6 se reproduce la Tabla 7.2 pero ordenando los circuitos de acuerdo con el porcentaje de utilización de CLBs. Se puede observar que el porcentaje de utilización no decrece con el tamaño del circuito, sino con su dificultad para ser rutado, es decir, con el valor de *WN*.

Tabla 7.6 *Circuitos de ordenados de acuerdo con el porcentaje de utilización de CLBs (PU).*

CIRC	*#CLBs*	*#E/S*	*WN*	*PU*
s298	1931	10	6	62%
bigkey	1707	426	7	55%
tseng	1047	174	7	45%
dsip	1370	426	7	44%
diffeq	1497	103	7	37%
alu4	1522	22	8	17%
misex3	1397	28	9	15%
seq	1750	76	9	14%
apex2	1878	41	9	11%
apex4	1262	28	10	8%
ex5p	1064	71	11	6%

7.5 Conclusión

En muchos estudios de partición para sistemas Multi-FPGA se ignora la topología final hacia la cual está dirigida esa partición. Sin embargo, éste es un aspecto muy importante debido a que los recursos de interconexión en los sistemas Multi-FPGA son realmente escasos. En este capítulo, hemos presentado *PUR*, un método basado en la *Optimización Combinatoria Termodinámica* y en las *Regiones de Steiner Rectilíneas* para implementar la partición de circuitos en MFPGAs. *PUR* puede tratar dos orientaciones del problema: partición de circuitos dirigida a mallas de FPGAs predeterminadas, y la búsqueda de la malla de FPGAs que mejor se adapta a un circuito dado. Por medio de una optimización de la ubicación basada en dos etapas, se adecua la demanda de comunicación entre FPGAs a la capacidad realmente existente. Además, las señales de E/S globales del circuito se asignan de forma natural a las FPGAs periféricas que pueden soportarlas.

PUR ha sido aplicado a un conjunto de circuitos de prueba. Inicialmente se exploró un esquema de partición de 2x2 para todos los circuitos. En el proceso de optimización se ha logrado un importante decremento en el *conjunto de corte* para todos los circuitos. Para aquellos circuitos con valores *WN* medios (tseng, s298, dsip, bigkey), esta reducción ha sido suficiente para obtener interesantes incrementos en los porcentajes de utilización de CLBs. Para aquellos circuitos con valores altos de *WN* (ex5p, apex4, seq, apex2), la escasez de pines de E/S de las FPGAs fuerza todavía a la aplicación de esquemas de partición de grano más fino. Sin embargo, el *conjunto de corte* logrado para estos circuitos podría ser un excelente punto de partida para la aplicación de otras técnicas como la del *cableado virtual* [BTDHHA97].

Capítulo 8

Principales aportaciones de la tesis y futuras líneas de investigación

Las dos etapas del diseño físico más implicadas en la utilización equilibrada de los recursos de cableado de los sistemas basados en FPGAs son las fases de ubicación y rutado. Como hemos visto a lo largo de esta tesis, existen diversos factores que afectan a la calidad de las soluciones y al tiempo de cómputo. Entre ellos podemos destacar dos:

- **Método para la búsqueda de soluciones**

  La ubicación es un problema NP-completo, de forma que es una de las tareas que más tiempo consumen en el diseño de circuitos orientado a FPGAs. Es necesario, por lo tanto, recurrir a métodos de optimización combinatoria para su resolución. Puesto que el tamaño de las FPGAs comerciales varía en un rango cada vez más amplio, el método escogido debe ser suficientemente flexible como para adaptar la búsqueda de soluciones a la calidad exigida y a los tiempos de cómputo disponibles.

- **Criterios para evaluar la calidad de las soluciones**

  El algoritmo de optimización combinatoria escogido es guiado por una función de coste que evalúa la calidad de las soluciones exploradas. Es necesario establecer criterios de medida de la calidad de las ubicaciones. Estos criterios pueden ir desde rápidas estimaciones que permitan acelerar la búsqueda de soluciones, a medidas detalladas acerca de su calidad. En último término, la calidad de una ubicación se pone a prueba en la etapa de cableado. Puesto que no existe un único método para cablear, podemos decir que la calidad final de una ubicación está asociada, en parte, al método de cableado escogido.

Este trabajo de investigación ha ido dirigido a la obtención de métodos eficaces para el diseño físico de circuitos sobre sistemas basados en FPGAs, teniendo en cuenta estos dos factores clave que afectan a la calidad y al tiempo de cómputo. Para alcanzar ese objetivo, ha sido necesario cubrir diversas etapas, tanto en el diseño físico orientado a FPGAs como a sistemas Multi-FPGAs. A continuación resumimos las **principales aportaciones** de esta tesis:

1. Método de *Optimización Combinatoria Termodinámica*

Muchos de los problemas derivados del diseño físico sobre sistemas basados en FPGAs son problemas de tipo NP-completo. Los algoritmos de búsqueda probabilísticos permiten la aproximación hacia soluciones óptimas para algunos de estos problemas. Entre estos algoritmos destaca el *Enfriamiento Simulado* (ES) que realiza una búsqueda dirigida en el espacio de soluciones. Su naturaleza probabilística le permite, durante la búsqueda, aceptar temporalmente soluciones peores que la actual para escapar de los valores mínimos locales. El *ES* se puede aplicar en la resolución de múltiples problemas de optimización combinatoria, obteniéndose buenos resultados cuando se realiza un ajuste fino de los parámetros del algoritmo. Sin embargo, la resolución de muchos problemas de optimización en diseño físico, requiere la investigación de nuevas funciones de coste, lo que lleva asociado costosos ajustes de los parámetros del algoritmo. Por otra parte, la elección de una función de coste para la resolución de un problema de ubicación determinado, no sólo se realiza sobre la base de la calidad de los resultados que puede proporcionar, sino también en relación a la potencia de cálculo y tiempo disponibles. La mejora continua en la potencia de cálculo, permite afrontar nuevos desafíos con funciones de coste más complejas. De esta forma, las funciones de coste se convierten en elementos dinámicos que integran nuevos objetivos cuando la capacidad de cómputo permite afrontarlos. En este trabajo, para resolver con éxito los problemas de congestión de los sistemas basados en FPGAs, ha sido necesaria la consideración y puesta a prueba de numerosas funciones de coste. Estas investigaciones llevaron asociados costosos estudios experimentales para la

adaptación del *ES* a los nuevos objetivos. Por otra parte, para hacer viable el tratamiento de los problemas planteados se recurrió a diferentes estrategias. Entre ellas, la optimización en etapas hizo necesario el ajuste de los parámetros del algoritmo para soluciones parcialmente optimizadas, sin que se perdiera la calidad de éstas debido a la naturaleza probabilística de los métodos utilizados. Como resultado de estas investigaciones se desarrolló un primer método de ubicación adaptativo, la *Optimización Natural* [VLH2000]. Posteriormente, este método fue mejorado con la *Optimización Combinatoria Termodinámica*, un nuevo método para la resolución de problemas de optimización combinatoria basado en principios termodinámicos. *OCT* es un método de optimización flexible, donde los parámetros del algoritmo se ajustan de forma automática a distintos problemas y funciones de coste. En el capítulo 4, se presentó el método de *OCT* comparándolo con una herramienta que utiliza el *Enfriamiento Simulado* para resolver el problema de la ubicación sobre FPGAs. *OCT* se ha adaptado a los distintos problemas de ubicación tratados, mejorando el rendimiento proporcionado por una herramienta que utiliza el *ES*. Asimismo, la **optimización en varias etapas** con funciones de coste crecientes en complejidad, se vio también beneficiada con este método, al preservar la calidad de las soluciones parciales obtenidas entre las diversas etapas de la optimización.

2. Algoritmo *RSR*: aproximación polinómica *al Árbol de Steiner Mínimo Rectilíneo*.

En el capítulo 6 se ha desarrollado un método rápido de cableado de redes multiterminal, es decir, una aproximación polinómica ($o(n^2)$) al *Árbol de Steiner Mínimo Rectilíneo*. A este nuevo método de cableado le hemos denominado ***Regiones de Steiner Rectilíneas*** *(RSR)* debido a su forma de trabajo basada en la utilización temporal de regiones rectilíneas indeterminadas.

La velocidad de este algoritmo se puede aprovechar en un doble sentido. Por una parte, *RSR* es un algoritmo de cableado suficientemente rápido como para poder ser utilizado como medida, en lugar de estimación de la congestión, durante la optimización de la ubicación. Con esta nueva orientación, la ubicación se optimiza para una herramienta de cableado concreta. Así, se elimina la incertidumbre sobre el cableado en

la etapa de ubicación, lo que permite una mayor focalización del proceso de optimización hacia el objetivo de reducir la congestión. Por otra parte, *RSR* produce aproximaciones de calidad del *Árbol de Steiner Mínimo Rectilíneo*, siendo ése el objetivo principal de las herramientas de cableado. Por lo tanto, el algoritmo *RSR* se puede utilizar, en sí mismo, como base para una herramienta de cableado rápida. El uso de *RSR* en la etapa de la ubicación sobre FPGAs, ha permitido reducir la congestión local, y por tanto, el ancho de canal demandado [VLH98], respecto a los mejores resultados publicados hasta la fecha [BR97].

3. Método *PUR* para la partición de circuitos orientada a Sistemas Multi-FPGA con topología de malla

La manera natural de considerar las restricciones de comunicación de las Mallas de FPGAs puede ser una partición, ubicación global y rutado global simultáneos, de forma que en todo momento se conozca la demanda real de conexiones entre FPGAs.

En esta tesis, se ha desarrollado un método de partición de circuitos sobre las Mallas de FPGAs, basado en ***la ubicación y rutado globales* (*PUR*).** En este trabajo, la Malla de FPGAs se modela mediante una gran FPGA continua, donde los bordes de separación entre FPGAs quedan difuminados y mantenidos únicamente como una plantilla superpuesta. Utilizando las técnicas de ubicación y rutado simultáneos desarrollados para FPGAs individuales, se optimiza la ubicación de los bloques lógicos con el objetivo de que el número de cortes producidos en cada segmento de la plantilla, sea menor que la capacidad de comunicación entre FPGAs. El método PUR realiza, a diferencia de otros métodos, una partición realista basada en medidas de congestión. En el capítulo 7 se presentó el método *PUR*, y se aplicó a un conjunto de circuitos de prueba para evaluar la calidad de las soluciones que generaba. A pesar de las fuertes carencias de comunicación entre las FPGAs, en las optimizaciones llevadas a cabo para los circuitos de prueba, se han conseguido interesantes incrementos en los porcentajes de utilización de las Mallas de FPGAs [VLH99].

Para terminar, podemos señalar algunas de las posibles **líneas de investigación** que se abren a raíz de los resultados presentados en esta tesis:

- El problema del cableado de una red multiterminal en una métrica rectilínea es un problema NP-completo. El algoritmo de las *Regiones de Steiner Rectilíneas* ha proporcionado un método rápido para su resolución. Se puede entrever, que investigaciones sobre la ordenación inicial de los nodos sobre los que se aplica el algoritmo, podrían acortar las diferencias de longitud de sus soluciones respecto al valor óptimo.

- El método *PUR* se ha aplicado a sistemas Multi-FPGA con topología malla obteniendo grandes disminuciones en el conjunto de corte. La aplicación del método *PUR* a problemas de características similares, como el diseño físico de circuitos orientado a módulos multi-chip, podría proporcionar beneficios similares.

- A lo largo de la tesis, se ha puesto de manifiesto la flexibilidad del método de *Optimización Combinatoria Termodinámica.* Así, se ha aplicado con éxito a distintas funciones de coste. Debido a sus propiedades adaptativas y sencillez de aplicación, la *Optimización Combinatoria Termodinámica* nace como un método muy prometedor, de confirmarse su idoneidad para afrontar el amplio rango de problemas tratados en la actualidad mediante el *Enfriamiento Simulado.* Sin embargo, su confirmación requiere la aplicación del método a otros problemas, realizando estudios comparativos con otros algoritmos.

Referencias

[ACGR98] M. J. Alexader, J. P. Cohoon, J. L. Ganley, G. Robins, "Placement and Routing for Performance-Oriented FPGA Layout", VLSI Design: An International Journal of Custom-Chip, Simulation and Testing, Vol. 7, NO. 1, 1998.

[Act] Actel ProASIC home page, www.actel.com/products/proasic

[Agu84] J. Aguilar, "Curso de Termodinámica", Ed. Alhambra, 1984.

[AJK82] K. J. Antreich, F. M. Johannes, F.H. Kirsch, "Anew approach for solving the placement problem using force models, " ISCAS, pp. 481-486, 1982.

[AK89] E. Aarts, J. Korst, "Simulated Annealing and Boltzmann Machines, A Stochastic Approach to Combinatorial Optimization and Neural Compupting", John Wiley & Sons, 1989.

[Alt] Altera home page, www.altera.com

[AM98] J. C. Alves, J. S. Matos,"RVC - A Reconfigurable Coprocessor for Vector Processing Applications", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 258-259, IEEE Computer Society Press, April 1998.

[AP99] F. Amiot, E. E. Pissaloux," Design of an Adaptive Reconfigurable Router for Robotics Vision Systems", Reconfigurable Technology: FPGAs for Computing and Applications, Proc. SPIE 3844, pp. 162-172, SPIE -- The International Society for Optical Engineering, September 1999.

[AR96] M. J. Alexander, G. Robins, "New Performance-Driven FPGA Routing Algorithms", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, No. 12, December 1996, pp. 1505-1517.

[Bax99] M. Baxter,"ICARUS: A Dynamically Reconfigurable Computer Architecture", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 278-279, IEEE Computer Society Press, April 1999.

[BDHSW97] J. Burns, A. Donlin, J. Hogg, S. Singh, M. de Wit,"A Dynamic Reconfiguration Run-Time System", Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pp. 66-75, April 1997.

[BEKH94] T. Benner, R. Ernst, I. Koenenkamp, U. Holtmann, "FPGA Based Prototyping for Verification and Evaluation in Hardware-Software Cosynthesis", Lecture Notes in Computer Science, Vol. 849, 1994.

[Ber92] P. Bertin, D. Roncin, J. Vuillemin, "Programmable Active Memories: A performance Assesment", ACM FPGA, February 1992.

[BFKKZ94] P. Berman, U. Föβmeier, M. Karpinski, M. Kaufmann, A. Zelikovsky. " Approaching the 5/4-Approximations for Rectilinear Steiner Trees". LNCS 855, 60-71, 1994.

[BH94] S. Bade and B. L. Hutchings,"FPGA-Based Stochastic Neural Networks: Implementation", Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pp. 189-198, April 1994.

[BL2000] F. Barat, R. Lauwereins, " Reconfigurable Instruction Set Processors: A Survey", Proceedings 11th IEEE International Workshop in Rapid System Prototyping, pp. 168-173, Jun. 2000.

[BM94] N. W. Bergmann, J. C. Mudge,"An Analysis of FPGA-based Custom Computer for DSP Applications", Proc. ICASSP '94, p. II-513--II--516, April 1994.

[BM98] I. M. Bland, G. M. Megson," The SYstolic Array Genetic Algorithm, An Example of Systolic Arrays as a Reconfigurable Design Methodology", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 260-261, IEEE Computer Society Press, April 1998.

[BR96] S. Brown and J. Rose, "FPGA and CPLD Architectures: A tutorial", IEEE Design&Test of Computers, pp. 42-56, 1996.

[BR96b] V. Betz and J. Rose, "Directional Bias and Non-Uniformity in FPGA Global Routing", ICCAD, 1996.

[BR97] V. Betz and J. Rose, "VPR: A new Packing, Placement and Routing Tools for FPGA Research", International Workshop on Field Programmable Logic and Applications, 1997. http://www.eecg.toronto.edu

[BR98] V. Betz, J. Rose, "Effect of the Prefabricated Routing Track Distribution on FPGA Area-Efficiency". IEEE Trans. on VLSI systems. Vol. 6 NO. 3 Sep. 1998, pp. 445-456.

[Bre77] M. A. Breuer, "A Class of Min-Cut Placement Algorithms," Proceedings DAC, pp. 284-290, 1977.

[Bro92] S. Brown, "Routing architectures and algorithms for field programmable gate arrays". Ph.D.Thesis, Dept. of Electrical Engineering University of Toronto, Feb.1992.

[BRV92] S. Brown, J. Rose, and Z.Vranesic, "A detailed router for field programmable gate arrays", IEEE Transactions on Computer-Aided Design, vol 11, no.5, pp.620-628, May 1992.

[BS86] M. E. Johnson Ihor O. Bohachevsky, M. L. Stein,"Generalized simulated annealing for function optimization", Technometrics, 28(3), 1986.

[BS98] N. W. Bergmann , P. R. Sutton,"A High-Performance Computing Module for a Low Earth Orbit Satellite using Reconfigurable Logic", Field-Programmable Logic: From FPGAs to Computing Paradigm, pp. 416-420, Springer-Verlag, Berlin, August / September 1998.

[BT95] V. Beiu, J. G. Taylor,"Optimal Mapping of Neural Networks onto FPGA's --- A New Constructive Algorithm", Lecture Notes in Computer Science, Vol. 930, p. 822-??, 1995.

[BTDHHA97] J. Babb, R. Tessier, M. Dahl, S. Z. Hanono, D. M. Hoki, A. Agarwal, "Logic Emulation with Virtual Wires", IEEE Transactions on CAD of IC and Systems, no.6 Jun. 1997, pp 609-626.

[CEA95] H. A. Chow, A. Elnaggar, H. M. Alnuweiri,"Highly parallel signal processing on the virtual computer", Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable Computing, Proc. SPIE 2607, pp.42-53, SPIE -- The International Society for Optical Engineering, October 1995.

[Che94] C. E. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling," DAC, 1994, pp. 690-695.

[CKL95] T. A. Cook, H.-R. Kim, L. Louca,"Hardware acceleration of n-body simulations for galactic dynamics", Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable

Computing, Proc. SPIE 2607, pp.115-126, SPIE -- The International Society for Optical Engineering, October 1995.

[CLWL95] C-D Chen, Y-S Lee, A. C-H. Wu, Y-L. Lin, "TRACER-fpga: a Router for RAM-Based FPGA's", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 14, no. 3, March 1995.

[CME93] C. Chou, S. Mohanakrishnan, J. B. Evans,"FPGA implementation of digital filters", Proceedings of the Fourth International Conference on Signal Processing Applications and Technology, pp. 80-88, 1993.

[CP86] J. P. Cohoon, and W. Paris, "Genetic Placement," Proc. IEEE ICCAD, pp.422-425, 1986.

[Cro95] D. C. Cronquist, "Simultaneous Place and Route for Wire-Constrained FPGAs", Dep. of Computer Science and Engineering, Univ. of Washington, Technical Report, March, 1995.

[CSZ94] P. Chan, M. Schlag, J. Zien,"Spectral-based Multi-way FPGA partitioning",Technical Report, University of California, Santa Cruz, Jack Baskin School of Engineering, Number UCSC-CRL-94-44, November 1994.

[CSZ96] P. K. Chan, M. Schlag, J. Zien. "Spectral-Based Multi-Way FPGA Partitioning". IEEE Trans. on CAD of IC's and systems. Vol. 15 NO. 5 May 1996.

[CW98] T. J. Callahan, J. Wawrzynek,"Instruction-Level Parallelism for Reconfigurable Computing", Field-Programmable Logic: From FPGAs to Computing Paradigm", pp. 248-257, Springer-Verlag, Berlin, August /September 1998.

[DBTHHA94] M. Dahl, J. Babb, R. Tessier, S. Hanono, D. Hoki, A. Agarwal, "Emulation of the Sparcle Microprocessor with the MIT Virtual Wires Emulation System", IEEE Workshop on FPGAs for Custom Computing Machines, pp. 14-22, IEEE Computer Society Press, April 1994.

[DGI94] J. Darnauer and P. Garay and T. Isshiki, "A Field Programmable Multi-Chip Module (FPMCM)", IEEE Workshop on FPGAs for Custom Computing Machines, pp. 1-10, April 1994.

[DHY93] P. Duggan and R. Haycock ,T. York,"A Field-Programmable Multi-chip Module for Implementing Boolean Neural Networks". Oxford

International Workshop on Field-Programmable Logic and Applications, pp. 24-34, Abingdon EE&CS Books, August 1993.

[Dar95] J. Darnauer et al, "A Field Programmable Multi-Chip Module (FPMCM)", Proceedings of IEEE Sympowsium on FPGAs for Custom Computing Machines, pp.1-9, 1995.

[DKTC95] T. Drayer, W. King, J. Tront, R. Conners,"MORRPH: A MOdular and Reprogrammble Real-time Processing Hardware", Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pp. 11-19, April 1995.

[Dob92] I. Dobbelaere et al, "Field Programmable MCM Systems. Design of an Interconnection Frame, " IEEE Custom Integrated Circuits Conference, pp. 4.6.1-4.6.4, 1992.

[EMHB95] C. Ebeling, L. McMurchie, S.A. Hauck, S. Burns, "Placement and Routing Tools for the Triptych FPGA", IEEE Transactions on Very Large Scale Integration(VLSI)Systems, Vol. 3, no. 4, pp. 473-482. December 1995.

[EWJ95] S. S. Erdogan, A. B. Wahab, K. Jainandunsing,"Reconfigurable computing for multimedia DSP applications", Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable Computing, Proc. SPIE 2607, pp.2-9, SPIE -- The International Society for Optical Engineering, October 1995.

[Fel68] W. Feller, "An introduction to probability theory and its applications", John Wiley & Sons, Inc. 1968.

[FW98] W. Fang, C.-H. Wu. "Performance-Driven Multi-FPGA Partitioning Using Functional Clustering and Replication". 35th DAC Proceedings, San Francisco, June 1998.

[Gal94] D. Galloway, D. Karchmer, P. Chow, D. Lewis, J. Rose, "The Transmogrifier: The University of Toronto Field-Programmable System", CSRI Technical Report (CSRI-306). University of Toronto, 1994.

[Gan96] G. Ganapathy et al, "Hardware Emulation for Fimtopmañ Verification of K5", Proceedings of the Design Automation conference, pp. 315-318, 1996.

[Gat95] J. Gateley et al, "UltraSPARC™-I Emulation,"Proceedings of the Design Automation conference, pp. 13-18,1995.

[GG84] S. Geman, D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images", IEEE Trans. Pattern Analysis and Machine Intelligence 6,1984,pp.721-741.

[GN96] P. Graham, B. Nelson,"Genetic Algorithms in Software and in Hardware - A Performance Analysis of Workstations and Custom Computing Machine Implementations", EEE Symposium on FPGAs for Custom Computing Machines, pp. 216-225, IEEE Computer Society Press, April 1996.

[GN98] P. Graham, B. Nelson,"FPGA-Based Sonar Processing", Proceedings of the ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays (FPGA-98), pp. 201-208, ACM Press, February 22-24 1998.

[Gok90] M. Gokhale, B. Holmes, A. Kopser, D. Kunze, D. Lopresti, S.Lucas, R. Minnich, P. Olsen, "Splash: A Reconfigurable Linear Logic Array", International conference on Parallel Processing, pp. 526-532, 1990.

[Gok91] M. Gokhale et al, "Building and Using a Highly Parallel Programmable Logic Array", IEEE Computer, pp. 81-89, January 1991.

[GRSZ94] J. Griffith, G.Robins, J.S. Salowe, T. Zhang, "Closing the Gap: Near-Optimal Steiner Trees in Polynomial Time", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, No. 11, November 1994, pp. 1351-1365.

[Guc2000] S. Guccione, "List of FPGA-based Computing Machines", www.io.com/~guccione/HW_list.html

[Had75] F. Hadlock, "Finding a maximum cut of a planar graph in polynomial time". SIAM Journal of Computing, 4, no. 3:221-225, September 1975.

[Haj88] B. Hajec, "Cooling schedules for optimal annealing", Mathematics of Operations Research 13, 1988, pp. 311-329.

[Hall70] K. M. Hall, "An r-dimensional quadratic placement algorithm", Management Science, pp.219-229, 1970.

[Hau94] S. Hauck, "Multi-FPGA systems". Ph. D. Thesis. University of Washington, 1994.

[HBE98] S. Hauck , G. Borriello, C. Ebeling, "Mesh Routing Topologies for Multi-FPGA Systems", IEEE Transactions on VLSI systems, Vol.6, NO. 3, pp. 400-408, 1998.

[HBS98] J.-O. Haenni, J.-L. Beuchat, E. Sanchez, "RENCO: A Reconfigurable Network Computer", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 288-289, IEEE Computer Society Press, April 1998.

[Hid99] J. I. Hidalgo,"Graph partitioning methods for multi-FPGA systems and reconfigurable hardware using genetic algorithms", Graduate Student Workshop, pp. 357-358, 13 July 1999.

[HKH94] R. W. Hartenstein, R. Kress, H. Reinig,"A Reconfigurable Data-Driven ALU for Xputers", IEEE Workshop on FPGAs for Custom Computing Machines, pp. 139-146, IEEE Computer Society Press, April 1994.

[Hig69] D.W. Hightower, "A solution to the line routing problem on a continuous plane", Proc. 6th Design Automation Workshop, 1969.

[HOIW94] T. T. Hwang, R. M. Owens, M. J. Irwing, K. H. Wang, "Logic Synthesis for Field- Programmable Gate Arrays", IEEE Transactions on CAD of Integrated Circuits and Systems", Oct. 1994.

[HPLT98] I. Hidalgo, M. Prieto, J. Lanchares, F. Tirado. "A parallel gentic algorithm for solving the partitioning problem in Multi-FPGA Systems". Proc. of 3rd International meeting on vector and parallel processing. VECPAR 98, Porto, 1998.

[HRS86] M. Huang, F.Romeo, and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule for Simulated Annealing," ICCAD, 1986, pp. 381 - 384.

[HVW90] J-M. Ho, G.Vijayan, C.K.Wong,"New Algorithms for the Rectilinear Steiner Tree Problem". IEEE Transactions on CAD, pp.185-193, Feb. 1990.

[Hwa76] F. K. Hwang, "On steiner minimal trees with rectilinear distance. SIAM Journal of Applied Mathematics, 30(1):104-114, January 1976.

[HWGG93] H.-J. Herpel, N. Wehn, M. Gasteier, M. Glesner,"A Reconfigurable Computer for Embedded Control Applications", IEEE Workshop on

FPGAs for Custom Computing Machines, pp. 111-120, IEEE Computer Society Press, April 1993.

[HZ2000] M. I. Heywood, A. N. Zincir-Heywood,"Register Based Genetic Programming on FPGA Computing Platforms", Genetic Programming, Proceedings of EuroGP'2000, LNCS, Vol. 1802, pp. 44-59, Springer-Verlag, 15-16 April 2000.

[Ing89] L. Ingber, "Very fast simulated re-annealing," Mathematical Computer Modelling, 12, pp. 967-973 (1989).

[Ing96] L. Ingber, "Adaptive simulated annealing (ASA): Lesson learned," Control and Cybernetics, 25, pp.33-54 (1996).

[JTYCS98] J. Jean, K. Tomko, V. Yavgal, R. Cook, J. Shah," Dynamic Reconfiguration to Support Concurrent Applications", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 302-303, IEEE Computer Society Press, April 1998.

[KB95] R. Kuznar and F. Brglez,"PROP: A Recursive Paradigm for Area-Efficient and Performance Oriented Partitioning of Large FPGA Netlists", International Conference on Computer Aided Design, pp. 644-649, IEEE Computer Society Press, November 1995.

[KBHBKA98] J. R. Koza, F. H. Bennett III, J. L. Hutchings, S. L. Bade. M. A. Keane, D. Andre,"Evolving Computer Programs using Rapidly Reconfigurable FPGAs and Genetic Programming", FPGA'98 Sixth International Symposium on Field Programmable Gate Arrays, pp. 209-219, ACM Press, 22-24 February 1998.

[KBK93] R. Kuznar, F. Brglez, and K. Kozminski. "Cost minimization of partition into multiple devices". 30th ACM/IEEE DAC, pages 315-320, Dallas, Texas, June 1993.

[KDCA96] W. E. King, T. H. Drayer, R. W. Conners, P. Araman,"Using MORPH in an Industrial Machine Vision System", Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pp. 18-26, April 1996.

[KG95] M. Khosravipour, H. Grünbacher, "VHDL-Based Rapid Hardware Prototyping Using FPGA Technology", Field-Programmable Logic and Applications, pp. 218-226, Springer-Verlag, Berlin, August/September 1995.

[KGV83] S. Kirkpatric, C.D. Gelatt, and M.P. Vecchi,"Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671-680, May 1983.

[Kha99] M. A. S. Khalid, "Routing Architecture and Layout Synthesis for Multi-FPGA systems". Ph. D. Thesis. University of Toronto, 1999.

[KSJA91] J. M. Kleinhans, G. Sigl, F. M. Johannes, K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems , Vol. 10, NO. 3, 1991, pp. 356-365.

[KRWDP99] H. Kropp, C. Reuter, M. Wiege, T.-T. Do, P. Pirsch, "An FPGA-based Prototyping System for Real-Time Verification of Video Processing Schemes", Field-Programmable Logic and Applications, pp. 333-338, Springer-Verlag, Berlin, August/September 1999.

[KS2000] H. Krupnova, G. Saucier, "FPGA Technology Snapshot: Current Devices and Design Tools", Proceedings 11th IEEE International Workshop in Rapid System Prototyping, pp. 200-205, Jun. 2000.

[KVGO99] M. Kaul, R. Vemuri, S. Govindarnjan, I. E. Ouaiss,"An Automated Temporal Partitioning and Loop Fission approach for FPGA Based Reconfigurable Synthesis of DSP Applications", Proceeding on DAC, pp. 616-622, 1999.

[KZ97] M. Larpinski, A. Zelikovsky, "New Approximation Algorithms for the Steiner Tree Problems", Journal of Combinatorial Optimization, 1997.

[LADDRSS98] W. Luk, P. Andreou, A. Derbyshire, F. Dupont-De-Dinechin, J. Rice, N. Shirazi, D. Siganos,"A Reconfigurable Engine for Real-Time Video Processing", Field-Programmable Logic: From FPGAs to Computing Paradigm, pp. 169-178, Springer-Verlag, Berlin, August / September 1998.

[Lan95] S. Lan, "Architecture and Computer Aided Design Tools for a Field Programmabe Multi-Chip Module", PH.D. Thesis, Standford University, 1995.

[LB93] G. G. Lemieux, S.D. Brown, "A detailed routing algorithm for allocating wire segments in field-programmable gate arrays", 4th ACM/SIGDA Physical design Workshop, April 1993.

[LBH76] J. H. Lee, N.K. Bose, F.K.Hwang, "Use of Steiner's Problem in Suboptimal Routing in Rectilinear Metric".IEEE Transactions on Circuit and Systems", pp.470-476, Jul. 1976.

[LD93] P. Lysaght, J. Dunlop,"Dynamic Reconfiguration of FPGAs", More FPGAs, pp. 82-94, Abingdon EE&CS Books, 1993.

[Lee61] C. Y. Lee, "An algorithm for path connections and its applications," IRE. Trans. Electronics Computers, vol. EC-10, pp. 346-365, 1961.

[LGIRC98] D. M. Lewis, D. R. Galloway, M. V. Ierssel, J. Rose, P. Chow, "The Transmogrifier.2: A Million Gate Rapid Prototyping System", IEEE Transactions on VLSI systems, vol. 5, no. 2, pp. 188-198, June 1998.

[LGSV2000] P. Lakshmikanthan, S. Govindarajan, V. Srinivasan, R. Vemuri,"Behavioral Partitioning with Synthesis for Multi-FPGA Architectures under Interconnect, Area and Latency Constraints", Parallel and Distributed Processing, pp. 924-931, Springer-Verlag, Berlin, May 2000.

[LM86] M. Lundi, a. Mess, “Convergence of an annealing algorithm”, Mathematical Programming 34, 1986, 111-124.

[LS96] P. Lysaght, J. Stockwood,"A Simulation Tool for Dynamically Reconfigurable Field Programmable Gate Arrays", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 4(3), pp. 381-390, September 1996.

[LW97] Y-S. Lee, C.-H. Wu, "A Performance and Routability-Driven Router for FPGA's Considering Path Delays", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems", Vol. 16, NO. 2, Feb. 1997.

[LW99] H. Liu and D. F. Wong,"Circuit Partitioning for Dynamically Reconfigurable FPGAs", ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 187-194, ACM Press, February 1999.

[Luc] Lucent FPGA page, www.lucent.com/micro/fpga/index.html

[MBS95] R. Murgai, R. K. Brayton, A. Sangiovanni-Vincentelli, "Logic Synthesis for Field-Programmable Gate Arrays", Kluwer Academic Publishers, 1995.

[MC93] S. Monaghan, C. P. Cowen,"Reconfigurable Multi-Bit Processor for DSP Applications in Statistical Physics", Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pp. 103-110, April 1993.

[MDGS98] M. Migliardi, J. Dongarra, A. Geist and V. Sunderam," Dynamic Reconfiguration and Virtual Machine Management in the Harness Metacomputing System", Lecture Notes in Computer Science, Vol. 1505, pp. 127-134, 1998.

[Mei95] R. Meier, "Rapid Prototyping of a RISC Architecture for Implemention in FPGAs", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 190-196, IEEE Computer Society Press, April 1995.

[MGTG97] T. Mathews, S. C. Gibb, L. E. Turner, P. J. W. Graumann," An FPGA implementation of a matched filter detector for spread spectrum communications systems", Lecture Notes in Computer Science, Vol. 1304, p. 364-??, 1997

[MI93] S. Monaghan, J. E. Istiyanto,"High-level Hardware Synthesis for Large Scale Computational Physics Targetted at FPGAs", More FPGAs, pp. 238-248, Abingdon EE&CS Books, 1993.

[Mich94] G. De Micheli "Synthesis and Optimization of Digital Circuits",Stanford University, McGraw-Hill, 1994.

[MK98] R. Macketanz, W. Karl, " JVX - A Rapid Prototyping System Based on Java and FPGAs", Field-Programmable Logic: From FPGAs to Computing Paradigm, pp. 99-108, Springer-Verlag, Berlin, August / September 1998.

[MKBGHS2000] R. Maestre, F. J. Kurdahi, N. Bagherzadeh, M. Fernandez, R. Hermida, H. Singh, "A Framework for Reconfigurable Computing: Task Scheduling and Context Management", accepted for publication in the IEEE Transaction on VLSI Systems.

[ML99] G. McGregor, P. Lysaght,"Self Controlling Dynamic Reconfiguration Field-Programmable Logic and Applications", pp. 144-154, Springer-Verlag, Berlin, August / September 1999.

[MM99] T. Moeller, D. R. Martinez,"Field Programmable Gate Array Based Radar Front-End Digital Signal Processing", Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, p. n/a, IEEE, April 1999.

[MMFC98] J. M. Moreno, J. Madrenas, J. Faura, E. Canto,"Feasible Evolutionary and Self-Repairing Hardware by Means of the Dynamic Reconfiguration Capabilities of the FIPSOC Devices", Lecture Notes in Computer Science, Vol. 1478, p. 345-??, 1998.

[MO98] T. Miyamori, K. Olukotun,"A Quantitative Analysis of Reconfigurable Coprocessors for Multimedia Applications", Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, pp. 2-11, IEEE, April 1998.

[MT68] K. Mikami, K. Tabuchi, "A computer program for optimal routing of printed circuit connectors", IFIPS Proc., H47:1475-1478, 1968.

[MVB95] L. Moll, J. Vuillemin, P. Boucard, "High Energy Physics on DECPeRLe-1 Programmable Active Memory", ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 47-52, February 1995.

[MVG99] I.I. Mandoiu, V.V. Vazirani, J.L. Ganley, "A new Heuristic for Rectilinear Steiner Trees", Proc. IEEE ICCAD, Nov 1999.

[NR98] S. K. Nag, R. A. Rutenbar, "Performance-Driven Simultaneaous Placement and Routing for FPGA's", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17. NO. June, 1998, pp. 499-518.

[OG95] U. Ober, M. Glesner. "Multiway Netlist Partitioning onto FPGA-based Board Architectures". ICCD 95, pp 150-155.

[OHG96] U. Ober, H.-J Herpel, M. Glesner, "CAPpartx: Computer Aided Prototyping Partitioning for Xilinx FPGAs, a Hierarchical Partitioning Tool for Rapid Prototyping", Field-Programmable Logic: Smart Applications, New Paradigms and Compilers, pp. 106-115, Springer-Verlag, Berlin, September 1996.

[PAM2000] "Programmable Active Memories project at DEC PRL", http://www.research.digital.com/PRL/publications/pam.html

[PDBS98] P. Poire, M.-A. Cantin, H. Daniel, Y. Blaquiere. Y. Savaria," A Comparative Analysis of Fuzzy ART Neural Network Implementations: The Advantages or Reconfigurable Computing", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 304-305, IEEE Computer Society Press, April 1998.

[PT98] H. Ploog, D. Timmermann,"FPGA Based Architecture Evaluation of Cryptographic Coprocessors for Smartcards", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 292-293, IEEE Computer Society Press, April 1998.

[PV99] A. Pandey, R. Vemuri,"Combined Temporal Partitioning and Scheduling for Reconfigurable Architectures", Reconfigurable Technology: FPGAs for Computing and Applications, Proc. SPIE 3844, pp. 93-103, SPIE -- The International Society for Optical Engineering, September 1999.

[PWH99] M. Piacentino, G. van der Wal, M. Hansen,"Reconfigurable Elements for a Video Pipeline Processor", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 82-91, IEEE Computer Society Press, April 1999.

[Qui75] N. R. Quinn, "The placement problem as viewed from the physics of classical mechanics. Proceedings of the 12th DAC, pages 173-178, 1975.

[Quic98] Quickturn Design Systems, Inc., System Realizer product brief, 1998. http://www.quickturn.com

[RBB91] O. Rettig, U. G. Baitinger, T. Büchner, "Prototyping of a Wheel Slip Evaluating Circuit with FPGAs", FPGAs, pp. 437-442, Abingdon EE&CS Books, 1991.

[RGS93] J. Rose, A. El Gamal, A. Sangiovanni-Vincentelli, "Architecture of Field-Programmable Gate Arrays", Proc. of IEEE, July 1993, pp. 1013-1029.

[RKW90] J. Rose, W. Klebsch, and J. Wolf, "Temperature Measurement and Equilibrium Dynamics of Simulated Annealing Placement", IEEE Trans. on CAD, vol. 9, no. 3, March 1990.

[Rob] G. Robin, www.cs.virginia.edu/~robins/steiner.tar

[RPM91] G. Rosendahl, Tw. Paille, R. McLeod,"In-system Reprogrammable LCAs provide a Versatile Interface for a DSP-based Parallel Machine", FPGAs. Oxford International Workshop on Field-Programmable Logic and Applications, pp. 392-400, Abingdon EE&CS Books, August 1991.

[RSV85] J. S. Rose, W. M. Snelgrove, Z. G. Vranesic, "ALTOR: An Automatic Standard Cell Layout Program", Canadian conf. on VLSI, 1985, pp.169-173.

[RS93] K. Roy, C. Sechen, "A Timing Driven N-Way Chip and Multi-Chip Partitioner", International Conference on Computer-Aided Design, pp. 240-247, 1993.

[SBD87] P. Siarry, L. Bergonzi, and G. Dreyfus, "Thermodynamic Optimization of Block Placement", IEEE Trans. on CAD, vol. 6, no. 2, March 1987.

[SDJ91] G. Sigl, K. Doll, F. M. Johannes, "Analytical Placement: A Linear or a Quadratic Objective Function?", Proceedings of IEEE Design Automation Conference, 1991, pp. 427-431.

[SDP92] S. P. Sample, M.R. D'Amouer, T. S. Payne, "Apparatus for emulation of Electronic Hardware System", U.S. Patent 5,109,353, April 28, 1992.

[Sas93] T. Sasao,"Logic Synthesis and Optimization: FPGA design by generalized Functional Descomposition", Kluwer Academic Publishers,1993

[Sch95] P. Schulz,"Extending DSP-Boards with FPGA-Based Structures of Interconnection", Lecture Notes in Computer Science, Vol. 975, 1995.

[SH87] H. Szu, R. Hartley, "Fast simulated annealing", Physics Letters A 122, 1987, pp.157-162.

[Sha48] C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, 27, pp.379-623, 1948.

[She99] N. A. Sherwani, "Algorithms for VLSI Physical design automation," Kluwer Academic Publishers, 1999.

[SLLKBFM2000] H. Singh, G. Lu, M. Lee, F. J. Kurdahi, N. Bagherzadeh, E. Filho, R. Maestre, Morphosys: Case Study of a ReconfigurableComputing System Targeting Multimedia Applications", Design Automation Conference Proceedings (DAC'00), pp. 573-578, Los Angeles, California, May 2000.

[SM93] P. Shaw, G. Milne, "A Highly Parallel FPGA-based Machine and its Formal Verification", Lecture Notes in Computer Science 705, Springer-Verlag, pp. 162-173, 1993.

[SMP99] R. P. S. Sidhu, A. Mei, V. K. Prasanna,"Genetic Programming using Self-Reconfigurable FPGAs", Field-Programmable Logic and Applications, pp. 301-312, Springer-Verlag, Berlin, August / September 1999.

[Sou78] J. Soukup, "Faster maze router", Proceedings of DAC, 1984.

[SWG91] J. P. Singh, W.-D. Weber, A. Gupta, "Splash: Stanford Parallel Applications for Shared-Memory", Technical Report, Stanford University, Computer Systems Laboratory, Number CSL-TR-91-469, p. 33, April 1991.

[SRR98] A. A. Shosha, P. Reinhart, F. Rongen,"Reconfigurable PCI-BUS Interface (RPCI)", Field-Programmable Logic: From FPGAs to Computing Paradigm, pp. 485-489, Springer-Verlag, Berlin, August / September 1998.

[SS84] C. Sechen, A. Sangiovanni-Vincentelli,"The TimberWolf placement and routing package," in Proc. Custom Integrated Circuit Conf. (Rochester, NY), 1984, pp. 522-527.

[SS90] W. Swartz, C. Sechen, "New Algorithms for the Placement and Routing of Macro Cells", ICCAD, 1990, pp. 336-339.

[SS95] W-J. Sun, C. Sechen, "Efficient and Effective Placement for Very Large Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 14, NO. 3, 1995, pp. 349-359.

[TBD98] A. Touhafi, W. F. Brissinck, E. F. Dirkx,"Simulation of ATM Switches Using Dynamically Reconfigurable FPGA's", Lecture Notes in Computer Science, Vol. 1482, 1998.

[TBDHA94] R. Tessier, J. Babb, M-Dahl, s. Hanono, A. Agarwal, "The virtual Wires Emulation System: A Gate-Efficient ASIC Prototyping Environment", 2nd International ACM/SIGDA WorkShop on Field-Programmable Gate Arrays, 1994.

[Tri93] S. Trimberger, "A Reprogrammable Gate Arrays and Applications," Proc. of IEEE, July 1993, pp. 1030-1041.

[Tud95] M. Tudruj,"Look-Ahead Dynamic Reconfiguration of Link Connections in Multi-Processor Architectures", Parallel Computing: State-of-the-Art and Perspectives Proceedings of the Conference ParCo'95, 19-22 September 1995.

[VBB93] J. Varghese, M. Butts, J. Batcheller, "An Efficient Logic Emulation System", IEEE Transactions on Very Large Scale Integrated (VLSI) Systems, 1(2), pp. 171-174, June 1993.

[VBRSTB95] J. Vuillemin, P.Bertin, D. Roncin, M. Shand, H. Touati, P. Boucard, "Programmable Active Memories: Reconfigurable Systems Come of Age", IEEE Transaction on VLSI Systems, 1995.

[Vij90] G. Vijayan , "Partitioning Logic on Graph Structures to Minimize Routing Cost". IEEE Trans. on CAD. Vol. 9 NO. 12. Dec. 1990, pp. 1326-1334.

[VL98] J. De Vicente, J. Lanchares. "FPGA Probabilistic Placement Avoiding Routing Congestion by Evolution Programs", International ICSC Symposium on Enginnering of Intelligent Systems, Tenerife (Spain), Feb. 1998.

[VLH98] J. De Vicente, J. Lanchares, R. Hermida, "RSR: A new Rectilinear Steiner Minimum Tree Approximation for FPGA Placement and Global Routing", 24th EuroMicro Conference, Västerås (Sweden) 1998, IEEE Press, pp 192-195.

[VLH99] J. De Vicente, J. Lanchares, R. Hermida, "Placement Optimization Based on Global Routing Updating for System Partitioning onto Multi-FPGA Mesh Topologies", 9th International Workshop on Field Programmable Logic and Applications, Glasgow (U.K) 1999, LNCS 1673, pp 91-100.

[VLH2000] J. De Vicente, J. Lanchares, R. Hermida, "Adaptive FPGA Placement by Natural Optimisation", Proc. 11th IEEE International Workshop on Rapid System Prototyping, Paris (France) 2000, pp 188-193.

[Wau91] T. C. Waugh,"Field programmable gate array key to reconfigurable array outperforming supercomputers", Proceedings of the IEEE 1991 Custom Integrated Circuits Conference, p. 6.6/1-4, 1991.

[Wer97] M. Wermelinger,"A Hierarchic Architecture Model for Dynamic Reconfiguration", Proceedings of the Second International Workshop on Software Engineering for Parallel and Distributed Systems, pp. 243-254, IEEE, 1997.

[WH97] J. Woodfill, B. V. Herzen,"Real-Time Stereo Vision on the PARTS Reconfigurable Computer", Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pp. 201-210, April 1997.

[WK99] R. D. Williams, B. D. Kuebert,"Reconfigurable Pipelines in VLIW Execution Units", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 298-299, IEEE Computer Society Press, April 1999.

[WL86] D. F. Wong, C. L. Liu, "A New Algorithm for Floorplan Design", Proc. 23rd ACM/IEEE Design Automation Conference, 1986, pp.101-107.

[WLL88] D. F. Wong, H.W. Leong, C.L. Liu, "Simulated Annealing for VLSI Desing", Kluwer Academic Publishers, 1988.

[WM94] Y-L. Wu, M. Marek-Sadowska, "An Efficient Router for 2-D Field Progammable Gate Arrays", EDAC, 1994, pp. 412-4216.

[WM95] Y-L. Wu, M. Marek-Sadowska, "Orthogonal Greedy Coupling -A New Optimization Approach to 2-D FPGA routing", DAC, 1995, pp. 568-573.

[WM97] Y-L. Wu, M. Marek-Sadowska, " Routing for Array-Type FPGA's", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 16, no.5, May 1997.

[WSKR99] K. Weiss, T. Steckstor, G. Koch, W. Rosenstiel, "Exploiting FPGA-Features during the Emulation of a Fast Reactive Embedded System", International Symposium on Field Programmable Gate Arrays (FPGA`99), ACM/SIGDA, pp. 235-242, ACM Press, February 21-23 1999.

[WWZ98] D. M. Warme, P. Winter, M. Zacharisen, "Exact Algorithms for Plane Steiner Tree Problems: A computational Study, Technical Report DIKU-TR-98/11, Dept. of Computer Science,University of Copenhagen,1998.(ftp.diku.dk/diku/users/martinz/geosteiner-3.0.tar.gz)

[Xil] Xilinx home page, www.xilinx.com

[YNK96] T. Yamauchi, S. Nakaya, N. Kajihara,"SOP: A Reconfigurable Massively Parallel System and Its Control-Data-Flow based Compiling Method", IEEE Symposium on FPGAs for Custom

Computing Machines, pp. 148-156, IEEE Computer Society Press, April 1996.

[YSS97] H.-K. Yun, A. Smith, H. Silverman,"Speech Recognition HMM Training on Reconfigurable Parallel Processor", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 242-243, IEEE Computer Society Press, April 1997.